Sistema FIEB | SENAI CIMATEC
PELO FUTURO DA INOVAÇÃO
CENTRO UNIVERSITÁRIO SENAI CIMATEC

Engenharia de Controle e Automação

Trabalho de Conclusão de Curso

Estudo de implementação de modelo de tolerância a falhas em um robô para inspeção de linha de alta tensão.

Apresentada por: Gabriel da Silva Santos

Março de 2021

Gabriel da Silva Santos

Estudo de implementação de modelo de tolerância a falhas em um robô para inspeção de linha de alta tensão.

Trabalho de Conclusão de Curso apresentada ao Curso de Engenharia de Controle e Automação do Centro Universitário SENAI CIMATEC, como requisito parcial para a obtenção do título de **Bacharel em Engenharia de Controle e Automação**.

Área de conhecimento: Interdisciplinar

Orientador: Jorsiele Damasceno Cerqueira

Co-orientador: Oberdan Rocha Pinheiro

Salvador

2021

Agradecimentos

Gostaria de agradecer primeiramente a Deus por permitir que chegasse até aqui e a minha família, que sempre me apoiou em minhas decisões e dentro das possibilidades me deu todo o suporte que eu poderia precisar para me tornar o homem que sou hoje. Também, em especial, eu gostaria de agradecer a Mariana Paroli por ter sido o meu principal apoio durante os momentos mais difíceis desse processo e por ter contribuído tanto para que eu conseguisse chegar até aqui.

Gostaria de agradecer a todos os profissionais do BIR que me ajudaram com ideias e feedbacks para o desenvolvimento desse trabalho. Agradecimentos especiais a Tiago Souza, Téo e Leandro Nozela por ideias que foram de suma importância. Agradaço especialmente ao professor Oberdan por toda compreensão e ajuda durante o período de desenvolvimento desse projeto.

Agradeço ainda ao Centro Universitário SENAI-CIMATEC pelos anos de faculdade que passaram, por toda a infraestrutura disponibilizada, e acima de tudo aos professores e educadores, sem os quais não seria possível estar aqui. Agradecimentos especiais ao professor, Taniel Franklin, coordenador do curso de Engenharia de Controle e Automação por todo o apoio e tutoria em momentos decisivos desse percurso.

Por fim, gostaria de agradecer a minha Orientadora, a professora Jorsiele Damasceno, por ser uma figura de muita compreensão diante dos problemas que aconteceram até chegar aqui e por todo incentivo a buscar a minha melhora como profissional e pessoa.

Muito obrigado.

Resumo

Com a necessidade constante de mais confiabilidade e disponibilidade de equipamentos industriais se torna cada vez mais necessário garantir que equipamentos complexos possam entregar aquilo que se espera deles e para isso este trabalho apresenta um estudo para a implementação de técnicas de tolerância a falha em um robô para inspeção de linha de alta tensão, através da utilização de uma simulação numérica de uma manipulador com dois eixos para validação conceitual pela presença de conjuntos com a mesma configuração no robô de inspeção. A simulação foi elaborada através do MATLAB utilizando a biblioteca *robotics* para criar o modelo do manipulador e gerar a transformada inversa permitindo a movimentação do manipulador. Também foram implementados os conceitos de detecção por limite de tempo e reconfiguração através de *rollback*. A partir dos resultados foi possível perceber como a utilização de técnicas de tolerância a falha leva um sistema que se encontra em estado de falha para um ponto onde o seu funcionamento é possível, mas tem um desempenho inferior ao sistema sem a falha.

Palavras-chave: Confiabilidade, Tolerância a falhas, Detecção, Reconfiguração, MATLAB

Abstract

With the constant need for more reliability and availability of industrial equipment, it becomes increasingly necessary to ensure that complex equipment can deliver what is expected of them and for this purpose this work presents a study for the implementation of fault tolerance techniques in a robot for high voltage line inspection, using a numerical simulation of a manipulator with two axes for conceptual validation by the presence of sets with the same configuration in the inspection robot. The simulation was performed using MATLAB using the robotics library to create the manipulator model and generate the inverse transform allowing the manipulator to move. The concepts of time limit detection and reconfiguration through rollback were also implemented. From the results it was possible to see how the use of fault tolerance techniques takes a system that is in a state of failure to a point where its operation is possible, but has a lower performance than the system without the failure.

Keywords: Reliability, Fault Tolerance, Detection, Reconfiguration, MATLAB

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Siglas

FMECA ..	Failure Modes, Effects and Criticality Analysis
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LIDAR ...	Light Detection and Ranging
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
RMN	Redundância Modular com N módulos
RMT	Redundância Modular Tripla
ROS	Robot Operating System
SISO	Single Input Single Output
SLAM	Simultaneous Localization And Mapping
SOTA	State Of The Art

Capítulo Um

Introdução

Em um mundo com cada vez mais tecnologia, a forma de viver e interagir da sociedade tem adaptado-se às mudanças causadas pelos avanços científicos. A forma de se relacionar afetivamente, de se alimentar, de se locomover e acessar informações foi alterada por empresas como Google, Facebook, Apple, Uber, Amazon e outras gigantes tecnológicas que a todo momento buscam formas novas de impactar o cotidiano social com seus produtos inovadores (SILVA, 2016).

A percepção de uma sociedade tecnológica já é tratada pela indústria do entretenimento a muitas décadas, onde a sociedade é normalmente composta por pessoas sempre conectadas a redes de comunicação móveis, carros voadores e robôs inteligentes, como é o caso dos filmes IA (Inteligência artificial) e EU, Robô . Embora sob a ótica de Hollywood, essas representações se tornam cada vez mais próximas em um mundo onde a robótica tem avançado a passos largos rumo ao desenvolvimento de veículos autônomos, robôs móveis inteligentes e sistemas complexos de reconhecimento de padrões (MATARIĆ, 2014).

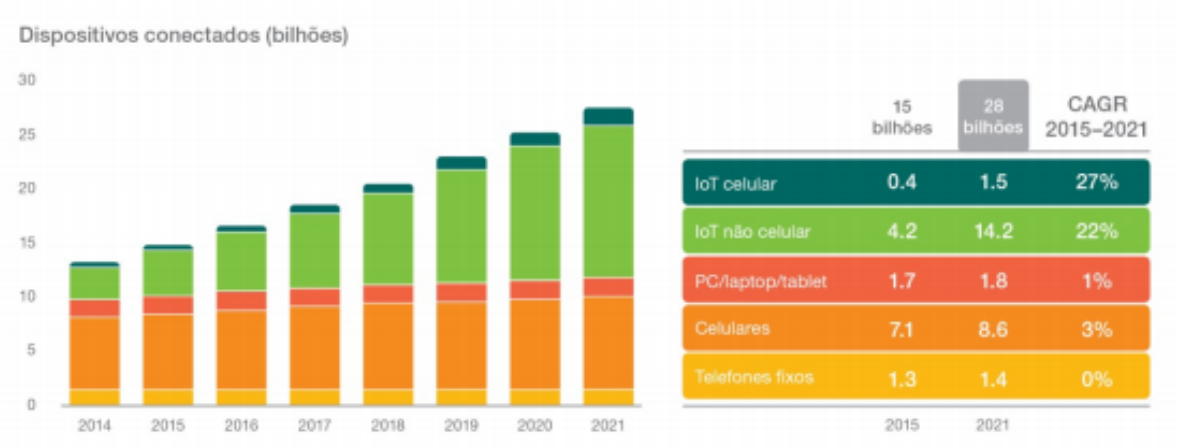


Figura 1.1: Gráfico de aumento de dispositivos conectados a internet.

Fonte: Ericsson, 2016

Esse caminho para um mundo mais conectado e com mais elementos não-humanos tem como objetivo um ganho considerável de confiabilidade e desempenho para as atividades que anteriormente eram executadas por pessoas. Esse aumento no uso de sistemas inteligentes para tomada de decisões críticas torna imprescindível que eles tenham a capacidade de entregar seus objetivos de forma rápida, prática e segura. Mas, até mesmo o sistema mais confiável ainda possui a possibilidade de falhar e por isso é necessário que ele tenha a capacidade de identificar o problema que está ocorrendo com ele e, caso possível após a

análise, tratar essa falha ou utilizar outro método para concluir a atividade com sucesso.

Neste sentido surgiu o conceito de tolerância a falhas para a aplicação em sistemas computacionais, que permitiu o desenvolvimento de redes mais estáveis e seguras. Esse aumento na disponibilidade dos sistemas fez com que outras áreas como sistemas embarcados, robótica e telefonia passassem a pensar seus desenvolvimentos com elementos capazes de suportar as possíveis anormalidades causadas por elementos internos ou externos dos seus produtos (WEBER, 2003).

1.1 Objetivos

Desenvolver uma simulação numérica de um manipulador com dois eixos e implementar técnicas de tolerância a falhas.

1.1.1 Objetivos Específicos

- Realizar levantamento de técnicas para tolerância a falhas;
- Realizar avaliação do Estado da arte das soluções de tolerância a falha para sistemas críticos;
- Gerar um manipulador com dois eixos em uma simulação numérica;
- Desenvolver sua movimentação através da cinemática inversa;
- Aplicar técnicas de tolerância a falhas para superar problemas mecânicos;
- Testar a precisão e otimização da simulação.

1.2 Justificativa

Observa-se no cenário mundial de avanços tecnológicos, a busca de implementar soluções para os problemas da produção que entreguem o melhor rendimento com o menor custo produtivo. Com isso, é perceptível o crescente aumento na necessidade de processos mais seguros e confiáveis. Os projetos de sistemas mecânicos, de software ou elétricos elaborados visam construir um conjunto que possua ou gere o mínimo de problemas possíveis, pois dependendo da aplicação um mau funcionamento pode custar muito dinheiro ou a vida de pessoas.

Os sistemas considerados críticos possuem um maior empenho no desenho e prototipagem para uma maior garantia de confiabilidade, para que ele entregue aquilo que foi projetado da forma esperada. Esses sistemas são mais complexos que a maioria por buscarem, através de teste incessantes de componentes, uma garantia de que as falhas não ocorrerão, entretanto por saberem que uma falha pode ser fatal eles possuem sistemas secundários ou mais que possam suportar a execução da tarefa caso o primeiro apresente problemas, como é o caso do sistema de trem de pouso retrátil de aeronaves que possui sistemas hidráulicos, pneumáticos e elétricos com multiplicidade para garantir um funcionamento pleno, como apresentado em (HANDBOOK-AIRFRAME, 2012).

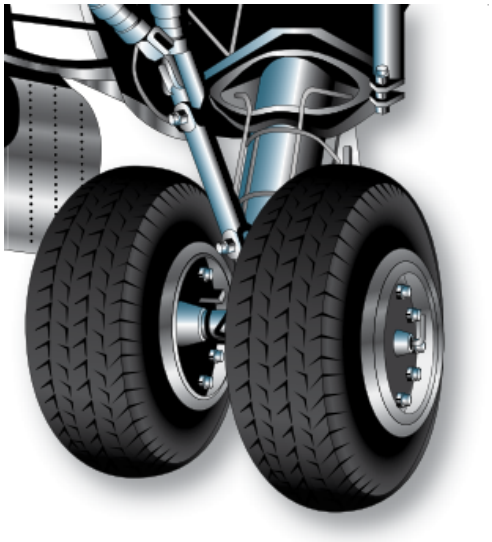


Figura 1.2: Trem principal duplo de um trem de pouso do tipo triciclo.

Fonte: (HANDBOOK-AIRFRAME, 2012)

Sistemas como esse que são desenvolvidos com a capacidade de suportar problemas são conhecidos como tolerantes a falhas. A concepção da tolerância a falha é criar sistemas que consigam executar suas missões mesmo na presença de falhas reduzindo o tempo em que o sistema estaria parado para manutenções e elevando o tempo de produção. Diversas áreas como computação, telefonia, energia nuclear e aeronáutica fazem uso dos conceitos da tolerância a falha para elaborar os seus projetos e outras áreas como a robótica tem começado a utilizar esse modelo de desenvolvimento.

Como é de conhecimento comum os robôs são compostos por diversos elementos como sensores, atuadores, processadores e por isso são considerados complexos pelas diversas relações que ocorrem entre os subsistemas. A sua complexidade para produção faz com que eles sejam caros para serem desenvolvidos e que, caso ocorram problemas, as manutenções também terão valores elevados por precisarem de uma mão de obra especializada. É nesse ponto em que a tolerância a falhas apresenta o seu valor, pois com as técnicas corretas é possível aumentar o tempo em que o robô está disponível, diminuindo as paradas para manutenções e aumentando assim o rendimento.

É importante frisar que o principal objetivo dos sistemas robóticos é permitir maior precisão e segurança para aplicações críticas ou em ambientes difíceis como o processo de inspeção de linhas de transmissão de energia elétrica. De acordo com (RANGEL; KIENITZ; BRANDÃO, 2009) esse trabalho por muito tempo foi feito por aeronaves que voavam próximo a fiação ou por meio terrestre, entretanto essas técnicas possuem diversos riscos de acidentes e são de alto custo.

Nesse contexto foram desenvolvidos robôs com o objetivo de diminuir os custos da inspeção e aumentar a segurança do processo. Como os apresentados em (DEBENEST et al., 2008) e (ALHASSAN et al., 2019), projetados para trabalhar em redes de alta tensão, que possuem grandes extensões e que estão expostas a intemperes como as chuvas e os ventos. A exposição a esse ambiente de trabalho pode gerar falhas ou defeitos que possam levar a não conclusão das atividades de inspeção, o que trás maiores custos ao processo, visto que cada missão mal sucedida é um gasto de recursos que poderiam estar sendo melhor aproveitados caso o sistema tivesse a capacidade de lidar com falhas inesperadas durante o exercício da missão.

Figura 1.3: Expliner: Robô para inspeção de linhas de transmissão.

Fonte: (DEBENEST et al., 2008)

Para tal, é proposto o desenvolvimento de um sistema de tolerância a falha para aplicação em robôs de inspeção de linhas de alta tensão. Esse sistema permitirá um ganho de confiabilidade para as missões, melhoria no uso dos recursos dispostos e a redução do retrabalho por missões fracassadas.

1.3 Organização do documento

Este documento apresenta 5 capítulos e está estruturado da seguinte forma:

- **Capítulo 1 - Introdução**: Contextualiza o âmbito, no qual a pesquisa proposta está inserida. Apresenta, portanto, a definição do problema, objetivos e justificativas da pesquisa e como este trabalho de conclusão de curso está estruturado;
- **Capítulo 2 - Fundamentação Teórica**: Apresenta a base teórica utilizada para o desenvolvimento do projeto;
- **Capítulo 3 - Materiais e Métodos**: Demonstra as etapas e equipamentos utilizados para a execução desse trabalho;
- **Capítulo 4 - Resultados**: Apresenta os resultados e etapas de execução dos testes;
- **Capítulo 5 - Conclusão**: Apresenta as conclusóes, contribuições e algumas sugestões de atividades de pesquisa a serem desenvolvidas no futuro.

Capítulo Dois

Fundamentação Teórica

Este capítulo dedica-se à definição de alguns conceitos sobre sistemas robóticos, bem como uma síntese do estado da arte e das técnicas para caracterização da tolerância a falhas.

2.1 Sistemas

Uma das noções mais importantes para a ciência é a visão sistêmica daquilo que está sendo analisado, ou seja, o conjunto e suas partes, como elas interagem entre si, como podem ser mais efetivas e como se degradam ao longo do tempo. Para um biólogo é importante compreender os processos biológicos pelos quais os seres vivos passam assim como para um físico é importante compreender a forma como os elementos da natureza influenciam os outros através das forças internas e externas presentes nessas relações.

A teoria que reage a correlação entre esses conjuntos é conhecida como Teoria Geral dos Sistemas, que foi elaborada como uma teoria biológica por Karl Ludwig Von Bertanlaffy, durante a década de 1920. A Teoria Sistêmica compreende organismos como compostos, ou seja, formados por uma totalidade mais abrangente que um simples somatório dessas partes (BERTALANFFY, 2008). A partir das análises de Bertanlaffy, é possível definir um sistema como "um conjunto de partes interagentes e interdependentes que, conjuntamente, formam um todo unitário com determinado objetivo e efetuam determinada função"(OLIVEIRA, 1999).

Para que um conjunto de partes possa ser considerado um sistema ele precisa possuir um propósito ou objetiva e a totalidade ou globalidade (SILVA; SANTOS; KONRAD,). O propósito ou objetivo é definido a partir da forma como as partes se organizam e a totalidade ou globalidade está relacionada a ideia de que caso ocorra um estímulo a alguma das partes isso irá afetar o sistema como um todo. Os sistemas podem ser classificados como:

- **Abertos x Fechados**: Sistemas abertos sofrem influência do ambiente externo enquanto os fechados não;
- **Concretos x Abstratos**: Sistemas concretos existem fisicamente, já os abstratos são modelos ou representações do mundo físico;
- **Naturais x Artificiais**: Sistemas naturais existem na natureza e os artificiais foram

criados pelo homem.

A compreensão da forma como um sistema é estruturado permite a análise das conexões entre seus elementos, dessa maneira o resultado do processo que ocorre em cada nó dessas conexões pode ser conhecido. Entretanto é possível que um sistema complexo possua conjuntos de nós ou processos que, caso sejam comprometidos, possam gerar problemas graves. A esses conjuntos é dado o nome de sistemas críticos.

2.1.1 Sistemas críticos

É possível definir sistemas críticos como sistemas sociotécnicos dos quais as pessoas ou negócios dependem (OLIVEIRA, 2009). Os sistemas sociotécnicos podem ser compreendidos como sistemas compostos por subsistemas sociais (pessoas, relações, habilidades,necessidades, etc.) e subsistemas tecnológicos (instalações, máquinas, equipamentos, tecnologia, etc.). Caso tais sistemas apresentem falhas ao executarem aquilo que foram planejados para fazer podem gerar grandes perdas econômicas, danos ambientais e até ameaça à vida humana (SOMMERVILLE, 2007), como ocorreu nos desastres de Bhopal na Índia e em Chernobyl na Ucrânia.

Sistemas como aeronáutico, nuclear, petroquímico, entre outros, podem ser classificados como críticos, pois eles necessitam de um alto nível de atenção em todas as etapas do seu processo, desde o projeto inicial até a manutenção dos equipamentos. Esses sistemas são compostos por diversos subsistemas que são potencialmente falhos e que possuem muitas relações de dependência entre si, o que faz com que seja vital ações preditivas, preventivas e corretivas visando manter o nível de risco o mais baixo possível (OLIVEIRA, 2009).

De acorco com (SOMMERVILLE, 2007) os sistemas críticos podem ser classificados como:

- **Sistema crítico de segurança**: Sua falha pode proceder em prejuízos, danos ambientais e perda da vida humana;
- **Sistema crítico de missão**: Sua falha pode ocasionar problema em alguma atividade conduzida a metas;
- **Sistema crítico de negócio**: Sua falha pode resultar em custos elevados para a empresa.

Quanto maior o nível de confiança no sistema mais caro ele se torna, pois para garantir esse nível de segurança do sistema são necessários custos elevados no projeto, desenvolvimento,

Figura 2.1: Usina nuclear Nogent-sur-Seine, na França.

Fonte: http://www.rfi.fr/br/franca/20111205-militantes-do-greenpeace-invadem-usina-nuclear-na-franca

validação e implementação do sistema. Com o conhecimento do elevado grau de risco que é inerente a operações com sistemas críticos os cientistas de diversas áreas tem se debruçado sobre o principal motivo que leva esses sistemas a desencadear acidentes de grandes magnitudes, com grandes prejuízos socioeconômicos, muitas vezes irreversíveis e com a possibilidade da interrupção da operação do sistema, as falhas. Esse esforço da comunidade científica permitiu a modernização e aprimoramento de mecanismos de supervisão e controle dos sistemas críticos. Esses avanços tem muita relação com a evolução dos sistemas e redes computacionais, porém as falhas continuam presentes nesses sistemas.

2.1.2 Falhas

Como apresentado na subseção 2.1.1 o maior problema para sistemas críticos são as falhas, pois nesses sistemas quando elas acontecem podem desencadear graves problemas, dependendo do nível e tipo da criticidade do sistema. Elas estão presentes em todos os tipos de sistemas e podem ser causadas por elementos internos e externos, por exemplo em um sistema robótico as falhas podem ser causadas por problemas no *software/hardware* (interno) ou por perturbações do ambiente (externo). Isso é compreensível pelo fato de que os componentes físicos envelhecem, os projetos de software sofrem com a falhabilidade humana no desenvolvimento das funcionalidades.

É importante compreender a diferença entre defeito, erro e falha, pois é nesse conceito

que se baseia o projeto a ser apresentado nesse documento. Um defeito pode ser definido como um desvio na especificação, quando um sistema opera de forma diferente daquilo que é esperado baseado no projeto original, por exemplo quando um sensor envia um valor diferente do esperado. O erro pode ser considerado um estado incorreto do sistema que pode causar um defeito, por exemplo um manipulador robótico que não consegue calcular corretamente a trajetória que deve percorrer. A falha é definida como a origem do erro (VALENTI et al., 2011).

O processo até o defeito pode ser exemplificado através do seguinte exemplo, um sistema robótico possui um problema na sua codificação (falha), que causará um erro no sistema no momento em que for executada e nesse momento poderá ser percebido pelo usuário quando o sistema apresentar respostas que fujam da especificação. Esse processo é mostrado na figura 2.2.

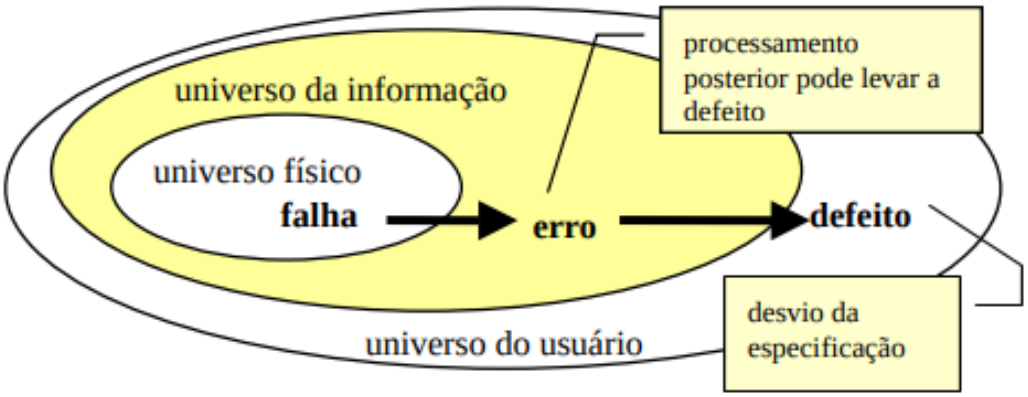


Figura 2.2: Relação entre falha, erro e defeito.

Fonte: (WEBER, 2003)

As falhas podem se propagar dentro de um sistema, por exemplo quando um um sensor envia um sinal errado para o sistema de controle que envia um sinal de atuação que não equivale ao esperado para a situação real. Nesse caso, na primeira é configurado um defeito sensorial e uma falha para o controle. Por conta disso é importante compreender:

- **Causa da falha**: O motivo original, sendo interno ou externo, que levou o sistema a não operar corretamente;
- **Natureza da falha**: Se a falha está no hardware, software, no projeto ou operação;
- **Duração ou persistência**: Se a falha é temporária ou permanente;
- **Extensão**: Qual a proporção do sistema a falha afeta, ou seja, se é modular ou global;
- **Valor**: Qual o impacto, ao longo do tempo, da falha no valor do sistema. Esse valor pode ser determinado ou indeterminado.

A partir da definição da estrutura da falha, desde a sua causa até o impacto no valor do sistema, é possível estimar a abrangência dela na capacidade do sistema entregar aquilo para que foi projetado, podendo assim identificar o quanto da qualidade do sistema foi afetada, ou seja, qual o resultado desses problemas na dependabilidade do sistema.

2.1.3 Dependabilidade

Um conceito importante de se compreender quando se fala de sistemas críticos é a dependabilidade, de acordo com (LAPRIE, 1992) ela é propriedade que define a capacidade de um sistema prestar um serviço em que se possa confiar. O serviço entregue por um sistema é o comportamento percebido pelo usuário, que pode ser humano ou não, quando ocorre uma interação.

Por se tratar de uma propriedade dos sistemas, ela é composta por alguns atributos que representam as diferentes leituras para aplicações. Quando se trata da continuidade do serviço é importante observar a confiança nele, quando se trata da prontidão para o uso é avaliada a disponibilidade, quando é esperada uma capacidade de evitar consequências problemáticas para o sistema é solicitado segurança e quando é observado a capacidade de prevenção de acesso não autorizado a informações é analisada a proteção do sistema (LAPRIE, 1992).

Na busca do desenvolvimento de sistemas com alta dependabilidade foram pensados e aplicados meios para que altos níveis de qualidade pudessem ser alcançados. Esses métodos utilizados podem ser classificados como:

- **Prevenção de falhas**: Como prevenir a ocorrência de falha no sistema;
- **Tolerância a falhas**: Como intervir para que o sistema entregue o serviço dentro das especificações, mesmo com a ocorrência de falhas;
- **Remoção de falhas**: Como reduzir a presença de falhas;
- **Previsão de falhas**: Como estimar a partir do que já aconteceu os futuros incidentes e as consequências das falhas.

As noções sobre dependabilidade até aqui apresentadas podem ser classificadas em três grupos, representados na figura 2.3.

- **Impedimentos para dependabildiade**: Falhas, erros e defeitos são indesejados,

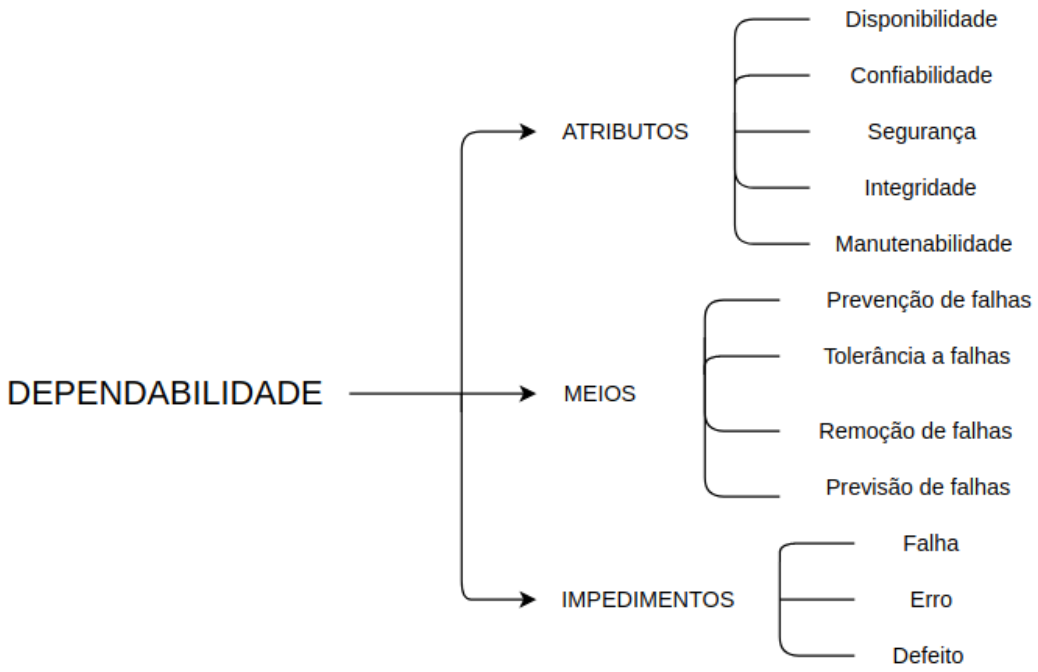


Figura 2.3: Árvore da dependabilidade.

Fonte: Adaptado de (LAPRIE, 1995)

porém são esperados para qualquer sistema. São causados por circunstâncias que muitas vezes não podem ser controladas;

- **Meios para a dependabilidade**: Prevenção de falhas, tolerância a falhas, remoção de falhas e previsão de falhas são métodos e técnicas que dão aos sistemas a capacidade de entregar serviços mesmo com a presença de falhas e permitem que o sistema tenha alta confiança nessa capacidade;
- **Atributos para dependabilidade**: Disponibilidade, confiabilidade, segurança, integridade e manutenabilidade são propriedades que quando analisadas refletem a capacidade do sistema de entregar serviços com qualidade.

2.1.3.1 Confiabilidade

Como apresentado anteriormente a confiabilidade é um dos atributos da dependabilidade e possui um papel importante na busca de um melhor desempenho dos sistemas. Ela é definida como a operação bem sucedida de um produto ou sistema, na ausência de quebra ou falhas (FOGLIATO; RIBEIRO, 2009). Entretanto quando se trata de uma análise voltada para a engenharia é necessário observar uma definição quantitativa que é proposta

por (LEEMIS, 2008).

"A confiabilidade de um item corresponde à sua probabilidade de desempenhar adequadamente o seu propósito especificado, por um determinado período de tempo e sob condições ambientais predeterminadas."

Nessa definição pode-se entender o item como sendo o elemento ou sistema a ser analisado. Essa percepção depende da quantidade de detalhamento esperado, pois, se o nível de detalhes exigidos for elevado, os itens analisados serão os elementos que compõem os subsistemas do objeto de estudo. Um exemplo que facilita a compreensão dessa definição é uma análise na confiabilidade de um sistema de condicionamento de ar, onde pode ser analisado como um único item (condicionador de ar) ou se o nível de detalhes for elevado serão analisados os seus subsistemas e os itens presentes neles como os motores, sensores e filtros.

Por se tratar de uma probabilidade tem-se que a confiabilidade deve representar um valor entre 0 e 1 e que ela está suscetível às relações probabilísticas. Por exemplo, se um sistema é composto por três elementos, a sua confiabilidade é a relação entre a multiplicação das confiabilidades dos elementos. Para o correto desenvolvimento de um modelo matemático que represente o desempenho de um elemento ou sistema é necessário que sejam estabelecidos parâmetros de desempenho, para que se possa avaliar se está sendo entregue um serviço dentro ou não daquilo que foi especificado. A representação mais simples e comum para itens é a binária, onde é estabelecido um ponto de corte que diz que o sistema não está mais apresentando funcionamento correto e entrou em modo de falha.

De acordo com (FOGLIATO; RIBEIRO, 2009) como a confiabilidade é medida através do tempo existem algumas implicações:

- O analista deve definir uma unidade de tempo para a realização de análises;
- Os modelos que descrevem os tempos até a falha utilizam a variável aleatória T para descrever o tempo;
- O termo tempo não deve ser interpretado literalmente, já que em muitos contextos o número de ciclos pode representar o tempo de falha do item;
- O conceito de confiabilidade deve ser associado a um período de tempo ou duração de missão;
- A determinação do que deveria ser usado para medir a vida de um item nem sempre é óbvia.

Como a confiabilidade possui uma análise quantitativa ela é atribuída através de medidas

que refletem valores para a estimativa. As medidas mais comuns são o tempo médio de reparo e tempo médio entre as falhas. O tempo médio entre falhas (MTBF em inglês) é o tempo médio em horas operáveis, utilizadas ou não, entre duas quebras consecutivas de um componente.

$$MTBF = \frac{TO}{NR} \tag{2.1}$$

Onde o TO é o tempo de operação do equipamento e NR o númenro de ocorrências corretivas. O tempo médio de reparo (MTTR em inglês) é o tempo médio gasto para repor a função do equipamento no caso de uma parada corretiva.

$$MTTF = \frac{TTR}{NR} \tag{2.2}$$

Onde TTR é o tempo total de reparo corretivo e NR o número de reparos corretivos.

2.1.3.2 Disponibilidade

A disponibilidade é definida como a capacidade de um item, mediante manutenção apropriada, desempenhar sua função requerida em um determinado instante de tempo ou em um período de tempo predeterminado (FOGLIATO; RIBEIRO, 2009). O conceito de disponibilidade é o mesmo caso a unidade seja não reparável e em unidades que podem ser reparadas os possíveis estados são em funcionamento ou em manutenção.

Uma vez que para aquele que solicita os serviços dos sistemas possui objetivos que devem ser alcançados num determinado período de tempo, qualquer estado atípico pode impactar no fato dessas metas não serem atingidas. Independente do sistema ou equipamento ter sofrido com uma falha ou estar em manutenção a sua inatividade gera prejuízos para os detentores dos bens.

É importante salientar que embora os bens de uma empresa envolvam diversos interesses, é necessário que exista um consenso entre as áreas para que todos os fatores que inabilitem o equipamento sejam levados em conta no momento do cálculo do indicador. Dessa forma a disponibilidade é dada por:

$$Disponibilidade = \frac{MTBF}{MTBF + MTTR} \tag{2.3}$$

2.2 Sistemas robóticos

Desde o início das civilizações o ser humano conseguiu operar diversos avanços quando começou utilizar as ferramentas corretas para cada situação. Com o passar do tempo e a contínua evolução das atividades de produção foi necessário utilizar máquinas que pudessem trabalhar de forma contínua e com isso ocorreram as revoluções industriais. Elas foram uma série de mudanças que substituíram o trabalho artesanal pela produção em série utilizando máquinas movidas a vapor (primeira revolução), derivados de petróleo e energia elétrica (segunda revolução)(HOBSBAWM, 2006).

A terceira revolução, que é conhecida como revolução tecnocientífica, trouxe uma nova integração entre ciência, tecnologia e produção. Com o foco na melhora dos processos já existentes e no desenvolvimento da ciência, a busca por meios mais eficazes de produzir foi o principal motor da grande modificação da sociedade nas última décadas. Assim, surgiram grandes avanços nos estudos da genética, redução entre as distâncias com meios de transportes mais eficazes e maior difusão de notícias e informações com a chegada de novos meios de comunicação como a televisão, fax e internet.

Nesse contexto também surge a robótica, como forma de acelerar processos produtivos utilizando os robôs, que conseguem executar as funções que antigamente eram feitas pelo homem de forma mais precisa, eficaz e sem a necessidade de descanso. Pela contínua necessidade de aumento na produção os robôs inseridos nos processos de produção tem se tornado uma realidade mais próxima, como é demonstrado na imagem 2.4.

Figura 2.4: Número de robôs insdustriais instalados no mundo (milhares).

Fonte: International Federation of Robotics

O estudo da robótica também vem ganhando espaço entre os jovens e se tornando parte da grade curricular na formação infantil, visto que com o crescimento na produção e utilização de robôs ao redor do mundo se torna cada vez mais necessário formar pessoas que estejam capacitadas para pensar um mundo que seja capaz de coexistir com esse nível

de tecnologia.

De acordo com (MATARIĆ, 2014) um robô é um sistema autônomo que existe no mundo físico e pode sentir o seu ambiente. Ele faz isso através dos sensores, que funcionam de forma similar aos sentidos humanos, que permitem que eles tenham noção do ambiente em que estão inseridos e a partir disso possam interagir com ele.

Figura 2.5: SpotMini e Atlas, robôs da Boston Dynamics.

Fonte: Boston Dynamics

Os sistemas robóticos autônomos podem ser divididos em duas áreas:

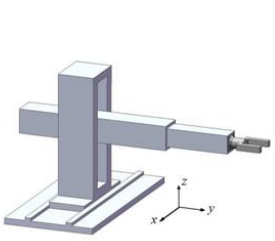
- **Robótica de manipulação**: Focada no desenvolvimento de robôs formados por juntas, que podem ser deslizantes, de rotação ou esféricas. Essa juntas são conectadas e a sua ação em conjunto permite movimentos de uma junta em relação a outra. A sua mobilidade depende da quantidade de juntas e articulações que ele possui. Um robô manipulador possui uma base fixa e o primeiro vínculo (junta) está preso a ela;
- **Robótica móvel**: Focada no desenvolvimento de robôs que possuem meios de se locomover dentro do ambiente em que estão inseridos e com isso executar missões em pontos distintos do espaço.

É importante salientar que a existência dessas áreas distintas não impedem o desenvolvimento de sistemas que fujam desses conceitos, como o desenvolvimento de manipuladores presos em bases móveis, o que mescla os conceitos principais das duas classes.

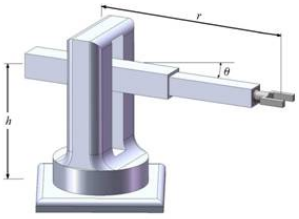
2.2.1 Robótica de manipulação

Como apresentado anteriormente, a robótica de manipulação é um ramo da robótica focado no desenvolvimento de robôs com o objetivo de manipular objetos ou executar funções repetitivas, como robôs utilizados em montadoras de veículos que fazem o papel de soldagem ou tiram e colocam as peças dos carros para serem utilizadas. Essa área da robótica ocupa a maior parte do mercado de robôs comerciais por ser a de maior demanda industrial, visto a necessidade de acelerar a produção pela sua capacidade de executar ações repetitivas em comparação aos seres humanos.

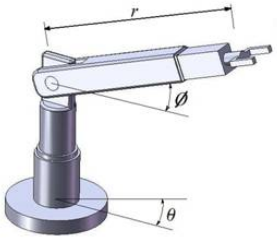
Existem diversas formas de classificar os manipuladores, mas a principal é pelas sua configurações que traz a cada tipo vantagens e desvantagens devido às suas geometrias. Quando se trata de atividades com alto nível de repetibilidade e sem limitações espaciais, a configuração cartesiana apresenta vantagens perante os outros, devido a sua movimentação linear. Quando a atividade necessita de alcance e a capacidade de ajustar a posição de chegada da ferramenta a configuração antropomórfica é a que melhor se adequa. Quando se trata de atividades com necessidade de levantamento de cargas, as configurações cartesiana e cilíndrica apresentam melhor resultado por poderem ser projetados para alta rigidez e capacidade de carga.



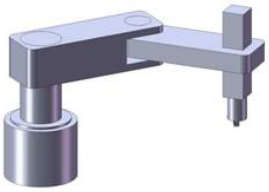
(a) Cartesiano



(b) Cilíndrico



(c) Esférica



(d) Scara

(e) Antropomórfico

Figura 2.6: Configuração de manipuladores

Fonte: (RIASCOS, 2010)

Como é possível notar, esses manipuladores são feitos para atividades específicas e cada configuração possui uma melhor adaptação a uma determinada atividade, seja ela mais precisa ou com maior necessidade de mobilidade ou repetibilidade. Para executar essas atividades é necessário que a controladora do robô faça o cálculo de transformação de um ponto para outro e isso precisa ser traduzido em movimentos lineares ou rotacionais para

que o robô saia do ponto A até o ponto B.

2.2.2 Robôs móveis

Como apresentado na seção 2.2 um robô é um sistema autônomo que utiliza os sensores para perceber o mundo a sua volta e, a partir dessas informações, pode interagir com ele. Quando se trata de robôs móveis a necessidade por informações aumenta, pois além de perceber aquilo que está a sua volta e interagir com o ambiente ele precisa saber em que posição ele está.

Um robô móvel pode ter diferentes meios de se locomover, podendo utilizar estruturas articuladas, esteiras, rodas ou pernas. A decisão da forma como o robô irá se locomover depende principalmente da atividade que ele deverá cumprir, das restrições impostas pelo ambiente, as limitações dos atuadores e a forma como ele será energizado (SILVA; MACHADO, 2001). A partir da decisão da forma como se mover, é possível definir a forma como a movimentação será percebida pelo robô através de sensores que estimam esses valores.

Figura 2.7: Warthog, robô terrestre anfíbio.

Fonte: Clearpath

Um dos principais métodos é a odometria, que utiliza encoders para calcular a movimentação das rodas. Ela é baseada na ideia de que o movimento circular das rodas pode ser convertido em deslocamento linear no chão. Esse método possui um erro associado pois podem ocorrer deslizamentos que não serão percebidos além de ter outras fontes de acúmulo de erro ao longo do tempo como um diâmetro desigual das rodas, desalinhamento e baixa resolução dos encoders (Nakju Doh; Choset; Wan Kyun Chung, 2003).

Para diminuir os erros da estimativa da odometria são utilizados sensores como a unidade de medição inercial (IMU) que é um dispositivo eletrônico que mede a velocidade angular, aceleração linear e orientação do corpo, usando uma combinação de acelerômetro, giroscópio

e magnetômetro (SICILIANO; KHATIB, 2016). Outro método que auxilia na diminuição do erro da odometria para ambientes externos é o sistema de posicionamento global (GPS em inglês) que através do sinal de satélites é estimada uma posição referente ao dispositivo conectado que possui um erro aproximado que depende do dispositivo que está conectado.

Associado a esses sensores que retornam valores estimados de posicionamento do robô existem métodos como o de localização e mapeamento simultâneo (SLAM em inglês), os quais utilizam as informações de odometria e as características extraídas do ambiente com câmeras ou sensor a laser (LIDAR em inglês). Essas técnicas permitem que, ao longo do tempo, o robô crie um mapa e que, ao passar por pontos conhecidos, ele possa se reposicionar nesse mapa, otimizando a sua localização e reduzindo a sua incerteza de posicionamento (Durrant-Whyte; Bailey, 2006). Existem diversas técnicas de SLAM 2D que é voltado para ambientes internos como o Gmapping (GRISETTIYZ; STACHNISS; BURGARD, 2005) que utiliza filtros de partículas para estimar de forma mais precisa, o Hector SLAM (KOHLBRECHER et al., 2011) que propõe um escaneamento robusto e uma rápida atualização do mapa e o Cartographer (HESS et al., 2016) que é um algoritmo desenvolvido pela Google para aplicações de SLAM em tempo real.

Por se tratar de um sistema complexo um robô móvel possui diversos subsistemas que permitem a ele executar suas funções, geralmente esses subsistemas são:

- **Percepção**: É o subsistema responsável por todo sensoriamento do robô. É ele que verifica o estado dos sensores e as informações que estão sendo recebidas;
- **Navegação**: É o subsistema responsável por toda movimentação do robô no espaço. É nesse sistema que é feita a tomada de decisão quanto a forma e o caminho que será tomado para chegar ao objetivo;
- **Persistência**: É o subsistema responsável pelo armazenamento dos dados obtidos pelo robô que serão necessários para as atividades;
- **Potência**: É o subsistema responsável pelo gerenciamento da energia do sistema.

Como é possível observar são diversos sistemas que possuem interação entre si, como é o caso da percepção em que os dados recebidos são enviados para a persistência para fazer o armazenamento para um possível pós-processamento, e por isso é necessário que exista um elemento responsável por gerenciar toda essa troca de dados de forma segura e otimizada.

2.2.3 Sistema Operacional Robótico

Assim como qualquer sistema computacional os subsistemas dos robôs necessitam de trocar informações para executar as suas funções vitais e, para isso, precisam que esses dados possam ser trocados da forma mais simples e rápida possível. É nesse contexto que surgem plataformas como o Sistema Operacional Robótico (ROS em inglês) com a intenção de facilitar o desenvolvimento da robótica.

O ROS é um *framework* para o desenvolvimento de sistemas robóticos que possui ferramentas, bibliotecas e convenções para simplificar a criação de sistemas complexos e robustos. Ele foi desenvolvido pela Open Robotics com o objetivo de permitir um ambiente de desenvolvimento colaborativo através de uma comunidade que possa compartilhar as bibliotecas produzidas com o intuito de acelerar a produção de conhecimento por evitar o retrabalho no desenvolvimento de pacotes. Ele possui uma arquitetura baseada em grafos onde o processamento se realiza através dos nós da rede que podem receber e enviar mensagens de diferentes tipos.

Ele trabalha com uma rede distribuída do tipo *peer-to-peer*, onde cada um dos nós da rede funciona tanto como cliente como servidor, que permite o compartilhamento de dados sem a necessidade de um controle centralizado. Os nós da rede comunicam-se através de tópicos, serviços ou parâmetros com mensagens que possuem formatos específicos para o tipo de atividade a ser exercida. Uma das grandes vantagens do ROS é que os nós podem ser desenvolvidos com qualquer tipo de linguagem de programação suportada como Python, C++ ou Java e isso não altera a forma como o sistema compreende as informações por conta da abstração que a rede do ROS apresenta.

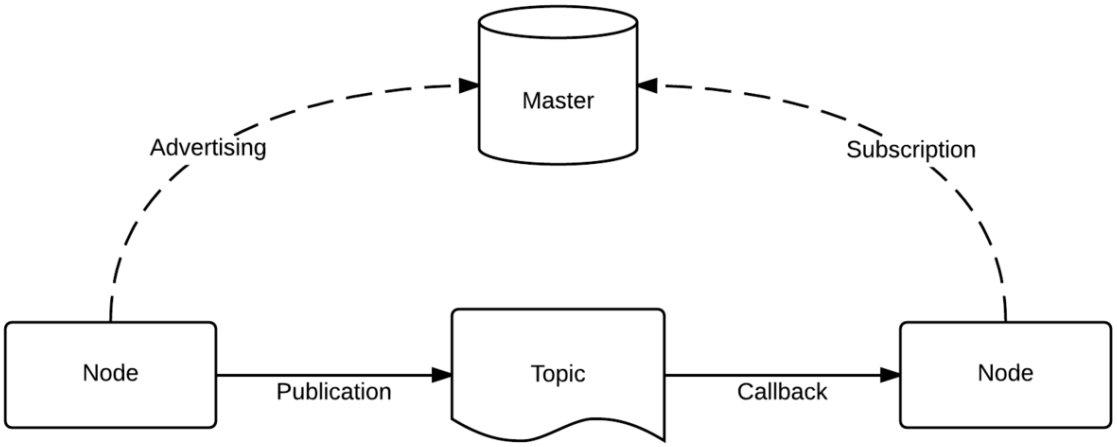


Figura 2.8: Esquema de funcionamento do ROS.

Fonte: Wikimidia

2.2.4 Simulação

Por robôs serem sistemas complexos e caros é importante ter certeza do que se pretende fazer e quais as respostas que serão obtidas a partir das intervenções que serão feitas. Sendo assim, é indispensável a utilização da simulação dentro do processo de desenvolvimento de uma solução com robôs. Dentro de uma simulação é possível testar os limites, criar cenários, testar teorias que num sistema físico poderiam gerar custos ou até se tornar inviáveis pela limitações que um elemento pesado ou grande pode trazer. Outra vantagem que a simulação traz para a robótica é a possibilidade da comparação entre soluções sem a necessidade de vários recursos físicos, ou seja, numa situação em que existam duas ou mais soluções possíveis a simulação permite o trabalho em paralelo dentro das organizações sem a necessidade de altos níveis de investimento com materiais e tendo assim resultados mais rápidos e satisfatórios.

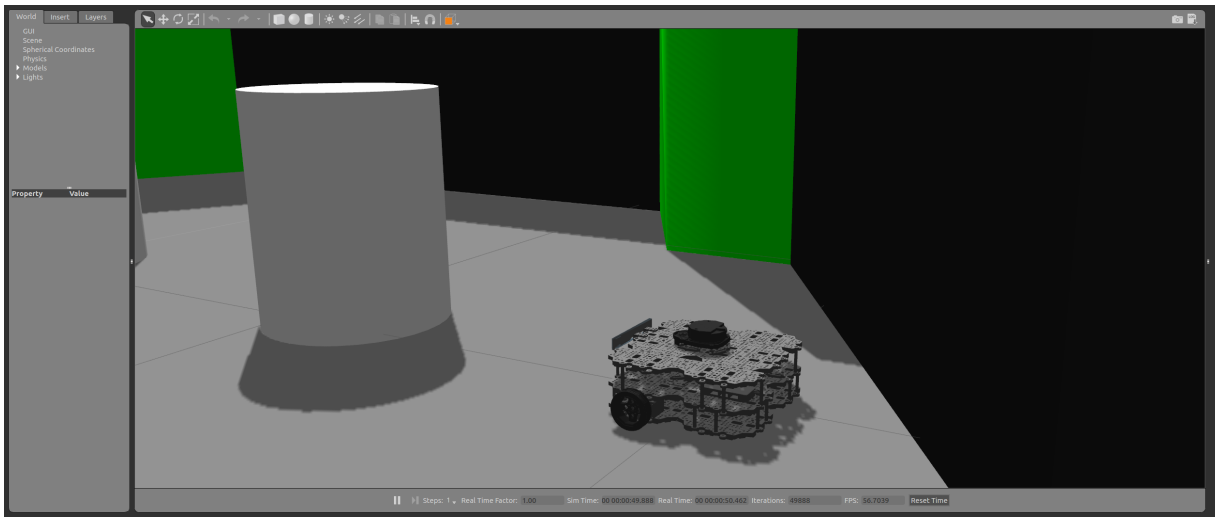


Figura 2.9: Turtlebot3 Waffle em um cenário no Gazebo.

Fonte: ROBOTIS e-Manual

O ROS também traz uma grande facilidade para o desenvolvimento de sistemas robóticos simulados por ter formas de se comunicar com diversos simuladores como a Unreal Engine 4 (SANDERS, 2016) que é uma ferramenta para o desenvolvimento de jogos muito utilizada atualmente por permitir um excelente tratamento da luz e um alto nível de detalhamento; o Webots (WEBOTS,) que um simulador robótico *open source* que permite a programação, simulação e modelagem dos sistemas; o V-Rep (ROHMER S. P. N. SINGH, 2013) que também segue a linha de ser um simular voltado pra robótica possui sensores, mecanismos, robôs e também permite a modelagem, além do Gazebo (AGUERO et al., 2015) que é o simulador nativo do ROS e que possui uma excelente simulação de física permitindo maior fidelidade da simulação com as respostas reais.

2.3 Tolerância a falhas

Como apresentado na subseção 2.1.3 a tolerância a falhas é um meio para se atingir a dependabildiade de um sistema através de intervenções que entreguem um serviço dentro daquilo que foi especificado e ela é definida por (AVIZIENS, 1976) como a capacidade que um sistema tem, sem a necessidade de assistência externa, de preservar a execução correta de seus programas e funções de saídas e entradas na presença de um certo número de falhas operacionais. O desenvolvimento de sistemas tolerantes a falhas são baseados em redundância, fazendo com que existam componentes adicionais ou algoritmos que são executados em situações específicas. As principais técnicas de tolerância a falhas fazem parte de dois grupos:

- **Detecção, localização e reconfiguração**;
- **Mascaramento**.

No mascaramento as falhas não aparecem como erros, pois são mascaradas. Nesse caso é empregado maior redundância por não diponibilizar tempo para fazer o procedimento de detectar, localizar e reconfigurar, por esse motivo é o mais indicado para aplicações críticas que precisam de respostas em tempo real. De acordo com (LEE; ANDERSON, 1990) o processo de tolerância a falhas possui quatro etapas, que quando ocorrem corretamente levam o sistema a sua maior disponibilidade.

Fases	Mecanismos
Detecção de erros	Duplicação e comparação Testes de limites de tempo Cão de guarda (watchdog timers) Testes reversos Codificação: paridade, códigos de detecção de erros, Hamming Teste de razoabilidade, de limites e de compatibilidades Testes estruturais e de consistência Diagnóstico
Confinamento e avaliação de danos	Ações atômicas Operações primitivas auto encapsuladas Isolamento de processos Regras do tipo tudo que não é permitido é proibido Hierarquia de processos Controle de recursos

Recuperação de erros	Técnicas de recuperação por retorno (backward error recovery) Técnicas de recuperação por avanço (forward error recovery)
Tratamento de falhas	Diagnóstico Reparo

Tabela 2.1: As quatro fases de Anderson e Lee

Com o objetivo de tolerar as falhas é necessário saber qual o estado do sistema e o que está acontecendo com ele. Sendo assim, é primordial identificar a falha corretamente. A primeira etapa de acordo com (LEE; ANDERSON, 1990) é a detecção do erro que pode ser feita através dos mecanismos listados na tabela 2.1. Um exemplo desses mecanismos é a duplicação e comparação, que utilizam duas peças idênticas e observam as respostas de cada uma aos mesmos dados, havendo incompatibilidade com os dados de saída o erro é sinalizado. Após a detecção de um erro mais de um sistema pode ser afetado, pois pode ocorrer uma latência entre a manifestação da falha e a detecção das suas consequências. Então antes de qualquer tentativa de lidar com o erro detectado é necessário avaliar até que ponto o sistema foi danificado.

Após a avaliação dos danos devem ser utilizadas técnicas para recuperação de erros, com o objetivo de levar o sistema de um estado incorreto a um estado definido e sem erros a partir do qual o sistema pode continuar operando dentro das suas especificações. A recuperação pode ser feita utilizando técnicas de retorno ou avanço que podem ser visualizados na figura 2.10.

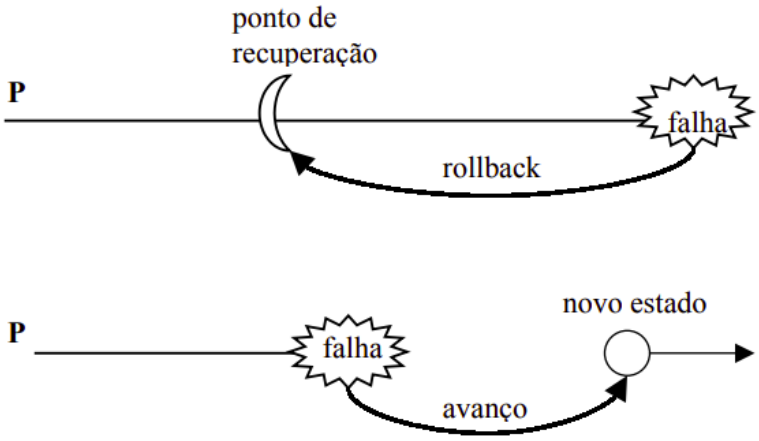


Figura 2.10: Recuperação de erro por retorno e avanço.

Fonte: (WEBER, 2003)

A técnica por retorno tem um efeito não desejado para sistemas complexos, pois a reversão da computação feita pode gerar um efeito dominó por conta dos processos que receberam e incorporaram mensagens do sistema. As consequências podem chegar a todos os sistemas fazendo com que retornem ao início do processamento. Por conta disso essa técnica não é indicada para sistemas em tempo real.

Mesmo que a fase de recuperação traga o sistema a uma zona livre de erros ainda pode ser necessário a aplicação de técnicas para que o sistema continue fornecendo o serviço dentro daquilo que foi projetado para fazer. O principal problema no processo de tratamento de falhas é que a detecção de um erro não serve necessariamente para identificar a falha que o originou, dessa forma o principal aspecto no tratamento de falhas é localizar elas com precisão para que a reparação e ou recuperação sejam efetivas. Nessa útima fase o tratamento das falhas pode ser divido em:

- **Localizar a origem do erro**;
- **Localizar a falha de forma precisa**;
- **Reparar a falha**;
- **Recuperar o resto do sistema**.

Geralmente é considerada a ideia de uma falha única e a partir disso é feita a localização dela. Ela é feita com um método de localização rápida, que é aplicada a um subsistema, e a localização fina, que determina o componente que está funcionando incorretamente. Para ambas é utilizado o diagnóstico por comparação dos resultados atuais e os esperados, que pode ser feito por um operador, local ou remoto, ou de forma automática. Com a falha localizada é feita a reparação através da remoção do componente danificado, podendo ser manual ou automático. O reparo automático pode ser feito por:

- **Degradação gradual**: Quando ocorre uma reconfiguração no sistema para operar com um número menor de componentes;
- **Substituição**: Quando o componente possui um substituto imediato disponível no sistema. Ela é utilizada em sistemas que possuem missões longas que não possuem a possibilidade de parada para o reparo, como é o caso dos satélites.

2.3.1 Redundância

Quando se fala de tolerância a falha, a chave para o sucesso está na redundância, a qual está presente em todas as técnicas de tolerância a falha de algum modo. Ela é definida por

(KOREN; KRISHNA, 2010) como a propriedade de possuir mais recursos que o mínimo necessário para o trabalho ser executado. A aplicação de redundância em técnicas de tolerância a falhas pode aparecer como:

- **Redundância de informação**;
- **Redundância temporal**;
- **Redundância de hardware**;
- **Redundância de software**.

É importante perceber que a utilização de redundância traz impactos para o sistema, pois pode aumentar o custo, modificação no desempenho original do projeto, aumento de tamanho ou peso e aumento no consumo de energia. Assim embora ela pareça a forma ideal de lidar com as falhas em um sistema resiliente, é importante definir qual a necessidade do sistema corretamente.

2.3.1.1 Redundância de informação e temporal

Na redundância de informação o que acontece é que junto com os dados do sistema são enviados sinais extras que servem para detecção de erros ou mascaramento de falhas. Um exemplo que é muito utilizado são os códigos de paridade, onde para cada n bits são enviados ou armazenados $n + 1$ bits. Esse dado extra indica se o número de bits com o valor 1 na mensagem é par ou ímpar, dependendo do tipo da paridade escolhida. Essa técnica serve para a detecção de falhas simples, que alteram apenas um bit da mensagem e por isso para falhas mais complexas são utilizados outros métodos como duplicação e códigos cíclicos. As técnicas de mascaramento utilizando a redundância de informação faz uso códigos de correção de erros como os códigos de Hamming, que são uma combinação de bits de paridade que faz tanto a detecção quanto a correção.

No caso da redundância temporal o que é feito é a repetição da computação dos dados, logo ocorre um aumento no tempo necessário para realizar um ciclo computacional e por isso é utilizado em sistemas onde o tempo não é um problema crítico. Pode ser tanto na detecção de falhas trasitórias quanto permanentes, entretanto o método utilizado é distinto para cada aplicação. Na detecção de falhas transitórias é feita apenas a repetição da computação e checagem dos resultados, caso existam resultados diferentes há um forte indicação de falhas. Já na detecção de falhas permanentes é feita uma repetição da computação com dados codificados e os resultados são decodificados antes da comparação com os resultados anteriores, dessa forma é possível identificar caso um estado anormal esteja ocorrendo.

2.3.1.2 Redundância de hardware

A redundância de hardware é baseada na utilzação de componentes físicos repetidos e pode ser passiva ou estática, que é utilizada no mascaramento de falhas, ativa ou dinâmica, utilizada no processo de detecção, localização e recuperação, e híbrida que é a combinação das outras duas com o objetivo de mascarar as falhas e possuir um sistema de longa duração, entretanto pode se tornar de alto custo de investimento.

No caso da redundância de hardware passiva os elementos do conjunto executam a mesma função e o resultado é determinado por um processo chamado de votação. Os exemplos mais comuns desse método são a redundância modular tripla (RMT) e redundância modular com n módulos (RMN). No caso do RMT ocorre um processo de votação entre os resultados do elemento que foi triplicado, podendo ser por maioria ou seleção de valor médio. Por conta do votador ser um ponto crítico do método, pelo fato de estar em série com os módulos que foram triplicados, é necessário que ele possua uma alta confiabilidade. Se o votador não possuir uma alta confiabilidade e se tornar um ponto de falha é preciso triplicar também o elemento votador ou fazê-lo em software.

A redundância de hardware ativa ou dinâmica é aplicada ao conceito de tolerância a falhas através das técnicas de detecção, localização e recuperação. Ela é utilizada em projetos onde os estados de erro não se mantêm por muito tempo e é usualmente preferível defeitos temporários do que aumentar consideravelmente os custos do desenvolvimento na busca de uma grande redundância para mascarar as falhas. Uma forma de implementação é utilizando módulos estepes, podendo serem alimentados ou não, que são elementos que podem substituir o elemento que apresenta a falha.

2.3.1.3 Redundância de software

Quando se trata de software, a redundância se apresenta de uma maneira bastante diferente do hardware, visto que um elemento de código defeituoso quando copiado vai apresentar o mesmo problema. É importante ressaltar que o objetivo durante o desenvolvimento do projeto de software é criar um sistema que não necessite utilizar técnicas de tolerância a falha, porém é muito difícil assegurar que um programa está livre de falhas. Para solucionar a tolerância a falhas em software, existem maneiras de aplicar a redundância para a detecção e mascaramento que não utilizam a replicação de programas, como:

- **Diversidade**;
- **Blocos de recuperação**;

- **Verificação de consistência**.

A diversidade, também chamada de programação diversitária, implementa diversas soluções para um problema e a resposta do sistema a esse teste ocorre por votação. Ela pode ser utilizada em qualquer etapa da construção de um programa, basta especificar o tipo de erro que se busca identificar. Essa técnica possui as vantagens de ter facilidade em identificar erros na fase de testes, tolerar falhas temporárias e permanentes e possível atuação em erros externos.

Blocos de recuperação é uma técnica similar a da diversidade, mas que utiliza programas secundários para a detecção de erros no programa primário utilizando um teste de aceitação para verificar se o processamento que ocorreu no programa é válido. A essência dessa técnica é testar os programas até que passe no teste de aceitação para que possa ser garantido uma execução livre de falhas.

Capítulo Três

Materiais e Métodos

Neste capítulo serão apresentadas as etapas para o desenvolvimento do trabalho. O capítulo é dividido entre uma apresentação do equipamento sobre o qual o estudo está sendo feito e a metodologia utilizada.

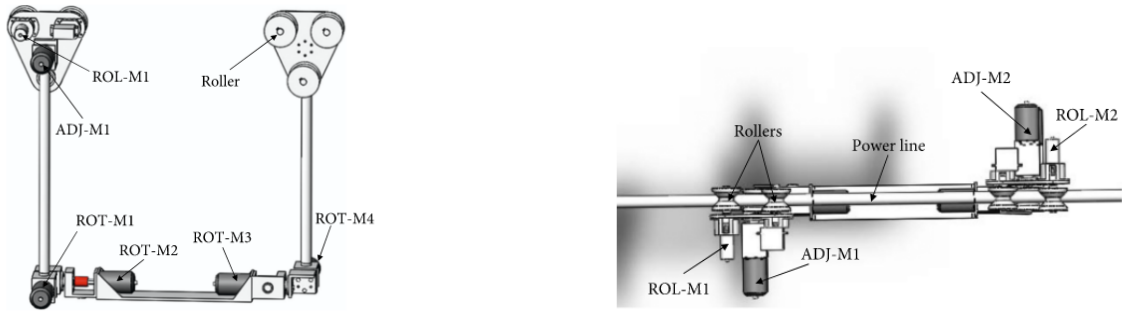
3.1 Robôs de inspeção

Como já foi apresentado em 1.2 o trabalho é focado em robôs autonômos para inspeção de linhas de alta tensão que tem como objetivo melhorar o processo de inspeção nessas linhas que tem altos custos para manutenção da forma que é feita atualmente, pois precisa deslocar um número considerável de pessoas e equipamentos, desde equipamentos para a inspeção propriamente dita até veículos de suporte como um helicóptero, dependendo da localidade. Como o Brasil é um país de escala continental a contínua necessidade de manutenção na vasta rede elétrica faz com que essa atividade se torne mais custosa ainda e permite que projetos como esse ganhem uma maior viabilidade econômica.

3.1.1 Estrutura mecânica

O robô que foi utilizado para o desenvolvimento do sistema apresentado no trabalho foi um modelo conceitual apresentado em (ALHASSAN et al., 2019) que foi pensado para ser um robô que se move apenas por uma linha de alta tensão que possui dois braços cilíndricos, duas garras triangulares e uma base retangular que comporta a bateria e os outros componentes eletrônicos. Cada garra possui três tambores para a movimentação, o inferior possui um motor que gera o torque a movimentação e os outros suportam o robô na linha.

A necessidade de se manter fixado a linha de transmissão enquanto executa a movimentação e inspeção traz consigo as dificuldade de enfrentar a resistência dos elementos naturais como o vento e a gravidade. Essa dificuldade fez com que um dos objetivos fosse construir um robô leve e que tivesse uma aerodinâmica que permitisse lidar com a resistência causada pelas rajadas de vento.



(a) Visão frontal.

(b) Visão superior.

Figura 3.1: Estrutura do robô de inspeção

Fonte: (ALHASSAN et al., 2019)

3.1.2 Movimentação

Para que o robô de inspeção de linhas pudesse efetuar a sua atividade, era necessário que ele fosse capaz não só de percorrer a linha de transmissão de forma linear, mas que pudesse passar pelos elementos construtivos que fazem parte da linha, como o grampo de sustentação. Para isso era necessário executar uma movimentação que pudesse passar pelo elemento sem que comprometesse a fixação na linha e para isso foi pensado num movimento utilizando os motores da base que pudesse criasse um desalinhamento entre as garras e permitesse a trasposição do obstáculo (ALHASSAN et al., 2019). Após isso, a extremidade mais próxima ao obstáculo solta as travas e os eixos se movimentam para que se crie uma distância entre o obstáculo e então ocorre um movimento linear para que essa parte ultraprasse o obstáculo e possa novamente ter as travas ativadas à linha e essa sequência se repete para a extremidade posterior.

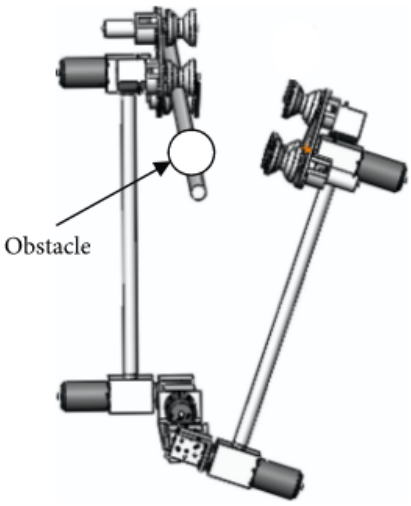


Figura 3.2: Transposição de obstáculo.

Fonte: (ALHASSAN et al., 2019)

3.1.3 Controle e Processamento

Por se tratar de uma atividade que tem uma necessidade de um controle preciso de posição, foi definido que o controle de cada junta fosse feito de forma independente, logo os comandos de controle de cada junta dependem da movimentação da própria junta. É importante salientar que esse tipo de controle descentralizado traz vantagens para o funcionamento do robô, pois as juntas não necessitam estar ligadas umas nas outras e isso reduz a probabilidade de problemas com comunicação e, para além disso, ocorre uma redução do custo computacional e do gasto com elementos para essa comunicação. Como trata-se de um sistema descentralizado, foi optada pela estratégia de controle SISO (*single input single output*) com uma entrada e uma saída para cada eixo das juntas do robô. Os modelos do tipo SISO são muito comuns e simples de controlar por se tratar apenas de uma variável de controle e monitaramento para alimentar o sistema.

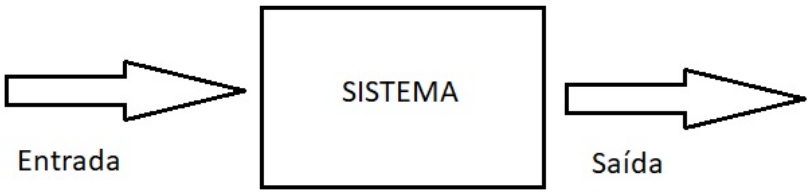


Figura 3.3: Controle SISO.

Fonte: Autoria própria

3.2 Metodologia

O trabalho desenvolvido foi feito dento do modelo de um Theoprax, que é uma método de trabalhos de conclusão de curso utilizado pelo SENAI Cimatec que tem como objetivo solucionar problemas práticos de empresas. Entretanto esse modelo não define um formato de metodologia específico para a execução. Dessa maneira, por se tratar de um projeto que está inserido na área da robótica, foi pensada uma metodologia baseada no modelo de desenvolvimento de projetos utilizado pela área de robótica do Cimatec. Esse modelo tem como base etapas bem estipuladas de estudo, design, prototipagem e testes com alguns marcos entre cada uma delas representados por entregas que conseguem sintetizar o que foi produzido durante aquela etapa.

Para metodologia desenvolvida foi utilizado um modelo exploratório com fontes de pesquisa secundárias e estruturado com três principais etapas de desenvolvimento: elaboração de ideias, desenvolvimento e testes. Dentro de cada etapa foram utilizadas formas de resumir o conhecimento que foi adquirido em entregas, similarmente ao proposto pelo modelo utilizado como referência. A representação da metodologia do trabalho pode ser observada

na figura 3.4.

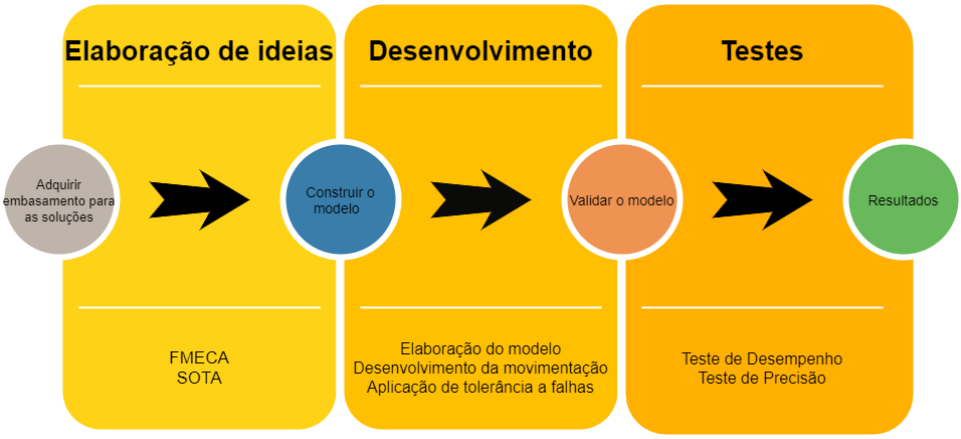


Figura 3.4: Metodologia.

3.2.1 Elaboração de ideias

Essa etapa de desenvolvimento teve como objetivo buscar, através de pesquisas, de qual forma a tolerância a falha poderia ser aplicada e se existiria viabilidade técnica e econômica para a aplicação. A etapa foi feita através da busca de referencial teórico sobre os dois principais assuntos que norteiam a pesquisa que é a robótica e tolerância a falha para que, com o conhecimento adquirido, pudesse ser mais simples pensar em meios de unir os dois temas em uma solução que englobasse o melhor dos dois mundos.

Após a familiarização com os temas, foi feita uma busca sobre como e onde é aplicada a tolerância a falha nos dias atuais e a partir dessa pesquisa foi elaborado um SOTA (State of the Art) que é um estudo do estado da arte, ou seja, uma busca por aquilo que de mais avançado existe de soluções com tolerância a falha nas diversas áreas da tecnologia. Através desse estudo foi possível verificar não só a imensa possibilidade de aplicações das técnicas, que estão presentes nas mais diversas áreas como telecomunicações, finanças, indústria, aeroespacial e computação, mas também a variedade de técnicas que são utilizadas para a solução de problemas em tempo real para evitar perdas. É importante salientar que esse estudo mostrou como é possível utilizar dos mesmos conceitos em diversas áreas para obter sistemas mais seguros e confiáveis, além de ter possibilitado visualizar sistemas complexos que jamais seriam possíveis sem as técnicas de tolerância a falha.

Alguns exemplos de aplicações de tolerância a falha que foram identificadas a partir desse estudo podem ser vistos em (TIAN et al., 2018) que é um controlador tolerante a falhas paramotores síncronos, pois, de acordo com os mesmos, os controladores lineares comuns apresentam baixa eficiência, após apresentarem as primeiras falhas, decorrentes das não linearidades que a corrente sofre e com isso um controlador que possa ter um rendimento

satisfatório precisa considerar esse problema sendo capaz de lidar com ele. Também em (ZHANG et al., 2018) é possível observar a aplicação para as telecomunicações onde se busca resolver problemas durante um processo de recuperação, etapa da técnica de detecção, recuperação e tratamento, onde, no período de redirecionamento do tráfego para o backup, o tempo necessário para o procedimento de recuperação seja mais longo do que o esperado,através de um modelo de roteamento paralelo com pacotes codificados que possuem um limite de transação de pacotes para acelerar o processo.

Com a compreensão das técnicas utilizadas e das aplicações mais inovadoras da tolerância a falha o próximo passo foi buscar como o sistema, no qual essas técnicas que seriam aplicadas poderia falhar, que no caso do trabalho seria o sistema mecânica do robô de inspeção e mais especificadamente nos motores Dynamixel, um modelo de servo motor. Para obter um entendimento mais seguro de como esse sistema poderia falhar e quais seriam os principais impactos disso para o funcionamento do robô, foi elaborado uma análise do modo de falha, efeitos e criticidade (FMECA em inglês) que é uma metodologia projetada para identificar modos de falha em potencial para um produto ou processo antes que os problemas ocorram, para avaliar o risco (BERTOLINI; BEVILACQUA; MASSINI, 2006). O modelo utilizado para o FMECA pode ser visualizado na figura 3.5.

Servomotor - Dynamixel										
Subsistema	Função	Modo de falha	Efeito da falha	Causa potencial	Controle atual	P.R.A				Ação corretiva recomendada
						S	P	D	R	
Potência	Enviar energia para o sistema	Não energização do sistema	Incapacidade de movimentação	Terminais queimados	Inspeção	9	5	5	225	Troca dos elementos
		Não continuidade na energização	Incapacidade de movimentação adequada	Mau contato entre terminais	Inspeção	7	5	3	105	Ajuste dos cabos
		Potência não suficiente	Incapacidade de movimentação adequada	Fuga de corrente	Análise de temperatura	4	5	4	80	Ajuste no aterramento

Figura 3.5: Modelo FMECA

Fonte: Autoria própria

Para fazer a análise do Dynamixel foi feita a divisão em quatro subsistemas principais: Potência, engrenagens, microprocessamento e comunicação. Em cada sistema foi colocada a sua função em relação ao conjunto e como essa função poderia ser impedida de forma parcial ou total através de uma falha. Para verificar o nível da criticidade de cada falha foi utilizado os parâmetros de severidade, probabilidade e detecção. A análise feita considerou que o R, que é o resultado da multiplicação entre os três parâmetros, abaixo de cem é razoável, entre cem e duzentos é de risco razoável e acima de duzentos é um risco elevado. A partir dessa análise foi possível compreender de que forma é possível evitar através de algumas possíveis falhas e outras que poderiam ser tratadas através das técnicas de tolerância a falha.

3.2.2 Desenvolvimento

Após a etapa de criação da ideia, onde foi adquirido o arcabouço necessário para a elaboração do estudo, foi feita a escolha de seguir um modelo de validação teórico a partir de uma simulação de um manipulador de dois eixos no MATLAB que pudesse, dentro de uma área de trabalho limitada, utilizar técnicas de tolerância a falhas para superar problemas mecânicos. Essa escolha foi feita por possuir diversas configurações de posição em que as técnicas poderiam ser testadas e validadas para a aplicação em robôs de inspeção e também foi feita em detrimento do objetivo inicial de desenvolver a pesquisa a partir da simulação de um robô de inspeção, utilizando ROS e Gazebo, e fazendo testes de mesa com os Dynamixels por conta das dificuldades causadas pela pandemia de SARS-COV2. O modelo inicial pode ser visualizado na figura 3.6.

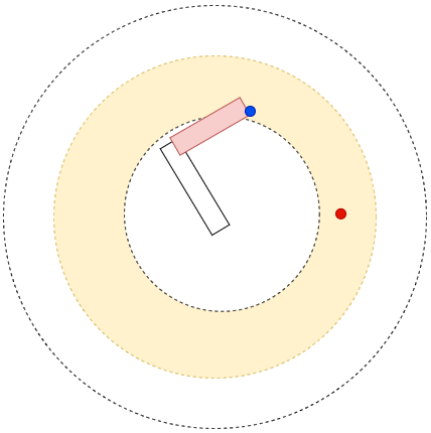


Figura 3.6: Modelo inicial do manipulador.

Dado o objetivo apresentado, o primeiro passo foi construir através do MATLAB um manipulador com dois eixos e para isso foi utilizada a biblioteca *robotics*. Ela é uma junção de métodos elaborados para robótica de manipulação e móvel que vão desde a criação da estrutura até as tomadas de decisões. Para criar o modelo do manipulador foi iniciada uma variável que recebe a estrutura da primeira haste que é criada pela função *rigidBody* e em seguida foi criada a sua junta sendo do tipo revolução.

Tendo a haste com uma junta criada foi necessário definir que seu vetor de translação seja convertido para uma transformação homogênea através da função *trvec2tform* e com isso é possível associar a junta criada a sua base. Para iniciar o robô como estrutura foi criada uma variável associada à função *rigidBodyTree*, a qual cria uma sequência de elementos que podem ser associados a partir de suas juntas, e com isso foi repetido o mesmo processo da primeira haste para a segunda e utilizado o comando *addBody* para definir em que ponto cada haste seria ligada ao robô, sendo a primeira ligada a base do robô,a segunda a extremidade da primeira e a ferramenta na extremidade da segunda. O resultado pode ser visto na figura 3.7 e 3.8.

```
Robot: (3 bodies)

 Idx    Body Name    Joint Name    Joint Type    Parent Name(Idx)    Children Name(s)
 ---    ---------    ----------    ----------    ----------------    ----------------
   1        link1        joint1        revolute            base(0)    link2(2)
   2        link2        joint2        revolute           link1(1)    tool(3)
   3         tool          fix1           fixed           link2(2)
```

Figura 3.7: Detalhes do robô.

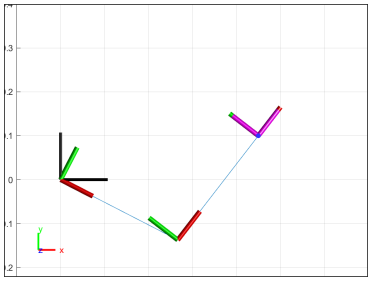
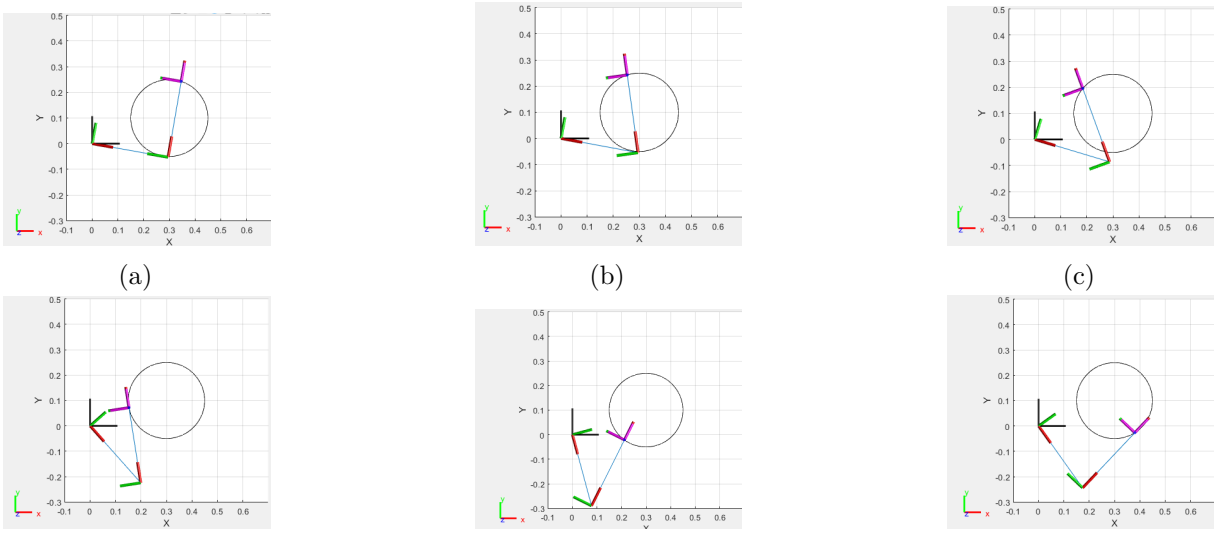


Figura 3.8: Manipulador com duas juntas.

Com a estrutura do manipulador montada, o próximo passo foi elaborar a movimentação do robô que foi feita através da utilização da cinemática inversa. A cinemática inversa busca através do conhecimento do posicionamento final do robô calcular a correta configuração das juntas que o proporcionaria. No caso do pacote *robotics* para o MATLAB a função que calcula isso é o *inverseKinematics* que, através do conhecimento da estrutura do robô, o seu *end effector* e a transformação homomogênea da posição alvo geram a movimentação. Para testar o método de movimentação do robô foi feita uma movimentação com o objetivo de gerar uma trajetória circular, para isso foi necessário criar uma estrutura de repetição que calculasse os pontos que o robô precisaria passar no círculo dentro de um período de vinte segundos. O resultado é possível observar através da figura 3.9.



(a) (b) (c)

(d) (e) (f)

Figura 3.9: Trajetória circular

Com o resultado da geração de uma trajetória circular, a etapa seguinte foi a geração de uma movimentação linear. Para gerar essa movimentação foi necessário alterar o cálculo

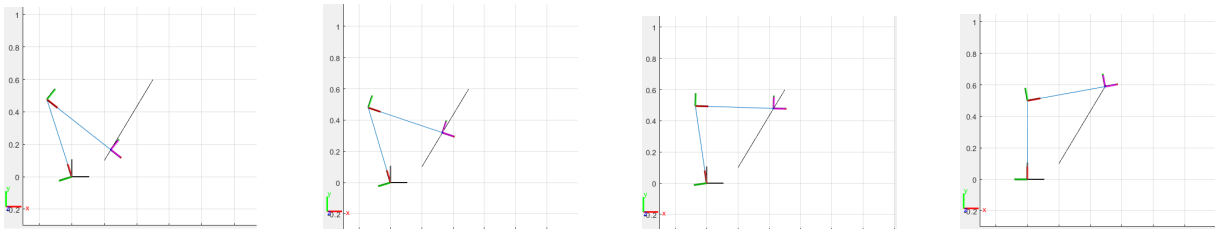
dos pontos, que para o movimento circular era feito pela equação 3.10 e para a linear foi feita pela equação 3.11. O resultado da movimentação pode ser visto na figura 3.12.

```
theta = t*(2*pi/t(end));
pointsc = center + radius*[cos(theta) sin(theta) zeros(size(theta))];
```

Figura 3.10: Equação de movimentação circular.

```
dx = (p2(1) - p1(1))/n; dy = (p2(2) - p1(2))/n;
pointsl = p1 +[dx dy 0].*t;
```

Figura 3.11: Equação de movimentação linear.



(a) (b) (c) (d)

Figura 3.12: Trajetória linear

Com a movimentação estabelecida, a próxima etapa do estudo foi elaborar uma forma de inserir uma limitação mecânica na simulação, entretanto o pacote *robotics* não possui uma função própria que insira falhas no modelo do robô elaborado. Com essa limitação foi pensado um meio de criar uma falha através de uma função que cria uma zona que impede a movimentação do manipulador em uma faixa angular específica para os eixos. O principal objetivo dessa falha é utilizar as técnicas de tolerância a falha para reconhecer, tratar e recalcular uma trajetória que não utilize a zona limitada pela falha mecânica e que possua o melhor resultado de performance. Essa zona pode ser visualizada na figura 3.13 sendo representada no formato de um quadrado.

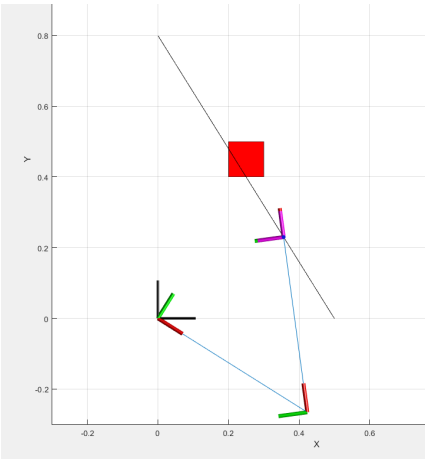


Figura 3.13: Zona de limitação de movimento.

A última etapa da fase de desenvolvimento foi utilizar métodos de tolerância a falha, já apresentados em 2.3, para que, mesmo com a presença de um impedimento mecânico, o manipulador possa concluir a sua missão. Foi escolhido que a estratégia implementada

seria a de detecção, localização e reconfiguração, pois tanto para o sistema simulado quanto para a implementação no robô de inspeção, seria inviável a utilização de redundância de hardware por conta das limitações físicas da aplicação e por isso não teriam meios para seguir com a estratégia de mascaramento, uma vez que a falha utilizada para o estudo é mecânica.

Cada uma das etapas da estratégia tem um método específico, já apresentados em 2.3, que foram escolhidos pela natureza da aplicação. Foi escolhido os testes de limites de tempo para a etapa de detecção por conta da possibilidade de uma atuação que vai reduzir o tempo de espera para cada etapa, pois uma vez conhecido o ponto final da atividade e o ponto em que ela está é possível saber quanto tempo será necessário para concluir cada movimento, sendo assim possibilitando que uma análise nessa variação possa verificar a existência ou não de uma falha. Por se tratar de um sistema simulado em que se conhece a origem da falha, não foi necessário utilizar um método de localização para a falha. Para a etapa de reconfiguração foi escolhido como meio a recuperação por retorno, ou seja o sistema retornará a um ponto anterior a falha e irá recalcular a trajetória considerando o problema encontrado. O fluxograma do processo pode ser visto na figura 3.14.

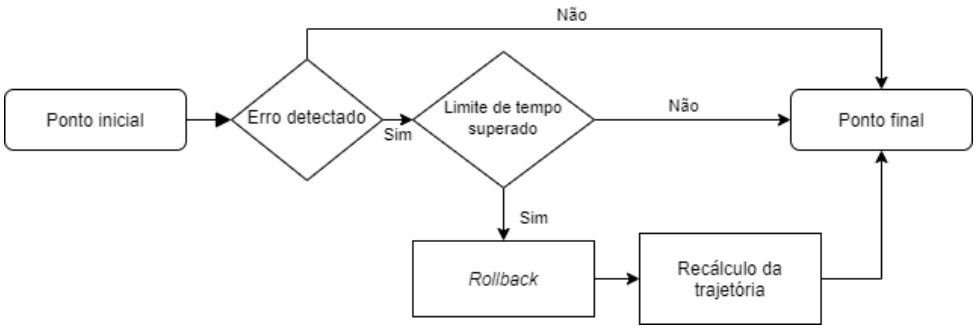


Figura 3.14: Fluxograma da estratégia de tolerância a falha.

3.2.3 Testes

Com o modelo pronto e com o funcionamento de acordo com o esperado o próximo passo para a conclusão da pesquisa foi elaborar como seriam feitos os testes e como seriam avaliados os resultados através deles. Foram definidas duas etapas de testes, sendo que cada etapa irá incluir tanto falhas intermitentes quanto falhas permanentes, para avaliar as respostas nas diferentes situações de indisponibilidade do sistema. Para a primeira etapa foi definido um teste para avaliar o tempo em que o sistema consegue concluir uma missão que consiste em definir um ponto inicial e um final, gerar uma falha no meio do percurso e perceber quantas iterações a mais serão necessárias para terminar o movimento, e ,para saber com exatidão em segundos, foi definido que cada iteração teria um quarto de segundo de duração. Um exemplo do teste pode ser observado na figura 3.15.

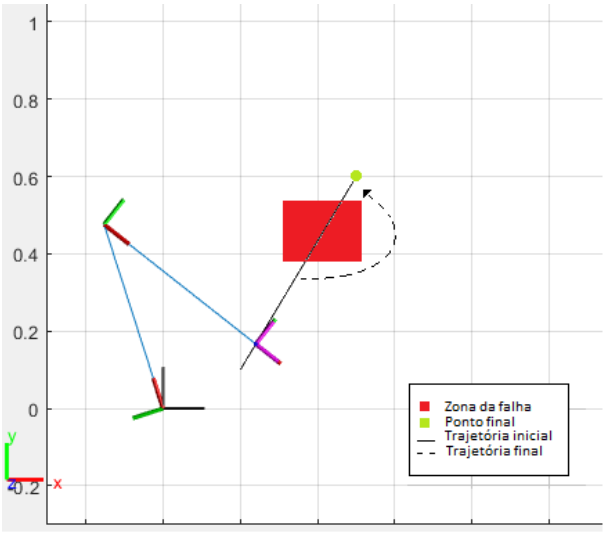


Figura 3.15: Exemplo do teste de desempenho.

O segundo teste tem como principal objetivo analisar as situações em que não foi possível chegar na posição solicitada na missão, pois dadas certas limitações mecânicas não existiriam configurações possíveis para o manipulador chegar ao objetivo, sendo assim, ele iria para o ponto mais próximo possível e nesse caso seria avaliado a proximidade entre o ponto indicado pela missão e o ponto final do manipulador. Um exemplo do teste pode ser observado na figura 3.16.

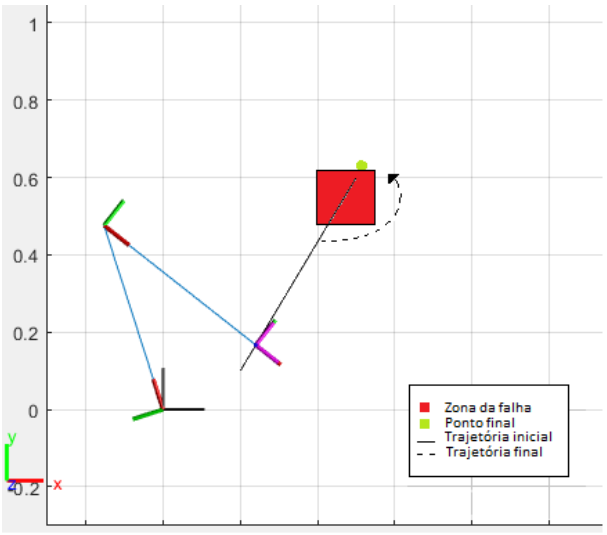


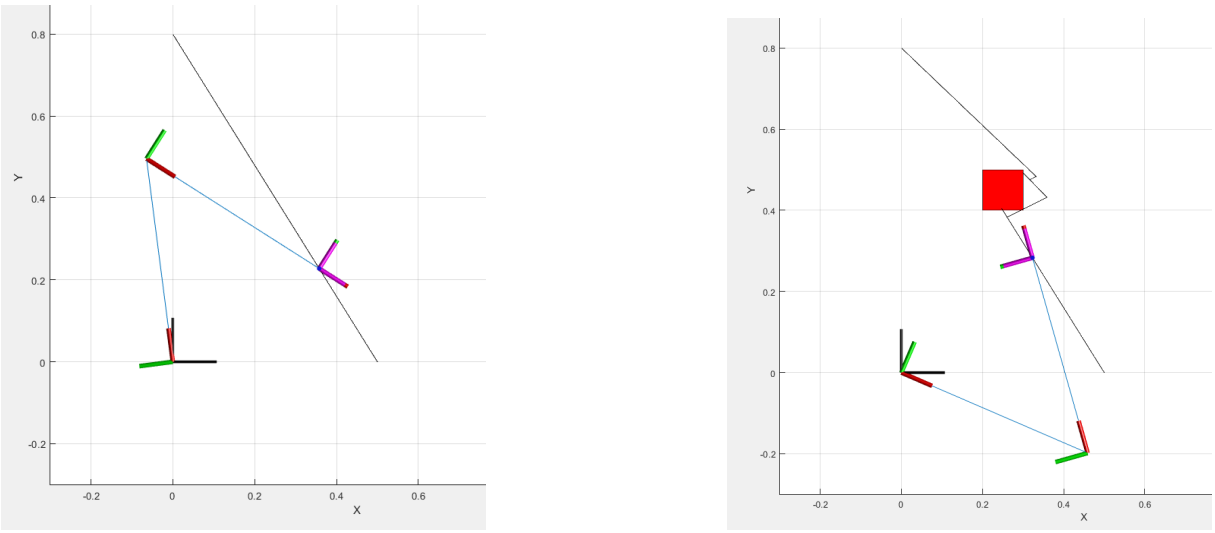
Figura 3.16: Exemplo do teste de precisão.

Capítulo Quatro

Resultados

Como apresentado no capítulo 2 é necessário que um sistema consiga entregar aquilo para o qual ele foi projetado, entretanto a existência de falhas levam os sistemas a condições de inoperabilidade parcial ou completa que afetam o desempenho e por consequência geram perdas financeiras, ecológicas ou até de vidas. Para lidar com esses sistemas que precisam manter a sua disponibilidade em detrimento do seu desempenho existem as técnicas de tolerância a falha. Os resultados obtidos a partir do desenvolvimento desse estudo consideraram verificar como o desempenho de um sistema é impactado pelas, visto que, mesmo com a implementação das técnicas, é importante para sistemas reais a contínua aplicação de rotinas para a manutenção e preservação dos equipamentos.

A partir do desenvolvimento apresentado no capítulo 3 foi possível elaborar um sistema que simula a movimentação de um robô com duas articulações dentro de um espaço de trabalho limitado a área da circunferência entre os raios de 0.5 e 0.8. Como dito anteriormente essa limitação visou criar cenários em que fossem mais visíveis a limitação de configurações possíveis para o manipulador se deslocar entre os pontos escolhidos, reduzindo assim a quantidade de soluções possíveis para situações adversas. Com simulação da movimentação linear foi possível inserir uma zona em que é verificada uma condição de falha que torna necessário a utilização das técnicas citadas em 2.3 que resultaram em um sistema capaz de retornar a pontos anteriores a falha simulada e recalcular a trajetórias para que aquela configuração faltosa seja evitada. Esse replanejamento pode ser visualizado na figura 4.1.



(a) Sem a falha

(b) Com a falha

Figura 4.1: Comparação entre as trajetórias.

4.1 Teste de desempenho

Na subseção 3.2.3 foi apresentado a estrutura em que os testes seriam executados. Para o teste de desempenho foi escolhido analisar o desempenho a partir do aumento iterações necessárias para executar a trajetória previamente definida dentro da área de trabalho do manipulador. Foram escolhidos pontos válidos e separados pelos quatro quadrantes do círculo unitário, sendo que os pontos do primeiro e segundo quadrantes foram espelhados para verificar para o terceiro e quarto respectivamente para verificar as variações que o mesmo deslocamento teria em sentidos diferentes. Inicialmente foi definido um ponto inicial e final e obtido a quantidade de iterações que seriam necessárias para executar toda a trajetória de ponto a ponto e em paralelo com a inserção da falha o sistema recalcula a trajetória aplicando a estratégia de tolerância a falha apresentada em 3.14 e entrega o valor de iteração que foi necessário para superar a situação problema. A tabela com os resultados pode ser visualizada na figura 4.2.

	TESTE DE DESEMPENHO					
	Ponto inicial	Ponto final	Nº iterações T1	Nº iterações T2	Diferença	Percentual
1º Quadrante	[0.5 0.0 0.0]	[0.0 0.8 0.0]	377	545	168	44,56%
	[0.6 0.2 0.0]	[0.1 0.6 0.0]	257	351	94	36,58%
	[0.5 0.5 0.0]	[0.0 0.5 0.0]	201	268	67	33,33%
	[0.6 0.25 0.0]	[0.2 0.7 0.0]	241	304	63	26,14%
	[0.6 0.2 0.0]	[0.2 0.7 0.0]	257	320	63	24,51%
	[0.5 0.3 0.0]	[0.0 0.8 0.0]	285	439	154	54,04%
	[0.0 0.8 0.0]	[0.6 0.4 0.0]	289	333	44	15,22%
	[0.2 0.7 0.0]	[0.6 0.4 0.0]	201	251	50	24,88%
2º Quadrante	[-0.6 0.2 0.0]	[-0.2 0.75 0.0]	273	352	79	28,94%
	[-0.6 0.2 0.0]	[0.0 0.8 0.0]	341	448	107	31,38%
	[-0.4 0.3 0.0]	[-0.2 0.75 0.0]	197	255	58	29,44%
	[-0.4 0.4 0.0]	[0.0 0.75 0.0]	213	278	65	30,52%
	[-0.4 0.4 0.0]	[0.0 0.7 0.0]	201	304	103	51,24%
	[-0.4 0.5 0.0]	[0.0 0.6 0.0]	165	291	126	76,36%
	[-0.6 0.0 0.0]	[0.0 0.7 0.0]	369	479	110	29,81%
	[0.0 0.7 0.0]	[-0.6 0.0 0.0]	369	469	100	27,10%
3º Quadrante	[-0.6 -0.2 0.0]	[-0.2 -0.75 0.0]	273	382	109	39,93%
	[-0.6 -0.2 0.0]	[0.0 -0.8 0.0]	341	468	127	37,24%
	[-0.4 -0.3 0.0]	[-0.2 -0.75 0.0]	197	333	136	69,04%
	[-0.4 -0.4 0.0]	[0.0 -0.75 0.0]	213	290	77	36,15%
	[-0.4 -0.4 0.0]	[0.0 -0.7 0.0]	201	317	116	57,71%
	[-0.4 -0.5 0.0]	[0.0 -0.6 0.0]	165	326	161	97,58%
	[-0.6 0.0 0.0]	[0.0 -0.7 0.0]	369	531	162	43,90%
4º Quadrante	[0.5 0.0 0.0]	[0.0 -0.8 0.0]	377	485	108	28,65%
	[0.6 -0.2 0.0]	[0.1 -0.6 0.0]	257	385	128	49,81%
	[0.5 -0.5 0.0]	[0.0 -0.5 0.0]	201	256	55	27,36%
	[0.6 -0.25 0.0]	[0.2 -0.7 0.0]	241	355	114	47,30%
	[0.6 -0.2 0.0]	[0.2 -0.7 0.0]	257	324	67	26,07%
	[0.5 -0.3 0.0]	[0.0 -0.8 0.0]	285	361	76	26,67%
	[0.0 -0.8 0.0]	[0.6 -0.4 0.0]	289	381	92	31,83%

Figura 4.2: Tabela do teste de desempenho.

A partir dos dados da tabela do teste de desempenho foi levantada a média e o desvio padrão de cada quadrante do círculo unitário e do total. Os resultados são apresentados na tabela abaixo.

Quadrante	**1**	**2**	**3**	**4**	**Total**
Média	32,41%	38,10%	54,54%	33,96%	**39,44%**
Desvio padrão	12,46%	17,27%	22,49%	10,17%	**17,65%**

Tabela 4.1: Médias e desvios padrão.

A partir desses resultados foi possível perceber que o terceiro quadrante foi o que apresentou a maior disparidade em relação a média geral e desvio padrão, sendo que os pontos utilizados nesse quadrante são os pontos do segundo quadrante espelhados. Uma situação específica foi visualizada em relação a trajetória entre [-0.4 0.5 0.0] e [0.0 0.6 0.0] no segundo quadrante e [-0.4 -0.5 0.0] e [0.0 -0.6 0.0] no terceiro quadrante, onde eles obtiveram os piores resultados entre os dados coletados sendo 76,36% e 97,58%. Esse tipo de resultado pode demonstrar uma deficiência do sistema em ajustar o seu posicionamento em relação ao eixo negativo da abscissa fazendo com que algumas trajetórias sejam recalculadas com uma eficiência menor do que o esperado.

4.2 Teste de precisão

Para o teste de precisão o procedimento para elaborar as situações problema foi relativamente distinto do teste de desempenho pela necessidade de impedir o sistema de encontrar uma solução para chegar ao ponto em que foi designado por conta da zona em que as falhas acontecem estar localizada no mesmo espaço que o ponto final desejado, logo era necessário que o sistema encontrasse a configuração mais próxima ao ponto desejado dentro daqueles que foram encontrados no momento em que foi verificado a incapacidade de encontrar uma solução com um erro entre ponto final esperado e real igual a zero. Como já dito foi colocada a zona em que as configurações são consideradas faltosas na mesma área em que o ponto final para fazer com que o sistema a partir de um número determinado de tentativas falhas encontre a solução com o menor erro e trace a trajetória até esse ponto. O cálculo do erro foi feito através da fórmula apresentada na figura 4.3 onde é feito o módulo da diferença entre o ponto atual representado pela variável *point* e o ponto final representado pela variável *p2*. Sempre que um erro menor é encontrado ele e o ponto referente são salvos e ao final de um número limitado de iterações caso uma solução com um erro igual a zero não seja encontrada é calculada a trajetória a partir do ponto com o menor erro.

```
erro = (norm(point-p2)/norm(p2))*100;
```

Figura 4.3: Equação do erro.

A partir dos dados do erro foi elaborada uma tabela com os resultados do teste de precisão com os mesmos pontos da tabela do teste de desempenho tendo como diferença os dados de erro e o ponto final com erro. A tabela pode ser visualizada na figura 4.4.

	TESTE DE PRECISÃO			
	Ponto inicial	Ponto final	Ponto final (real)	Erro
1º Quadrante	[0.5 0.0 0.0]	[0.0 0.8 0.0]	[0.0154 0.7564 0.0]	5,7753%
	[0.6 0.2 0.0]	[0.1 0.6 0.0]	[0.1730 0.5905 0.0]	12,1028%
	[0.5 0.5 0.0]	[0.0 0.5 0.0]	[0.0546 0.5063 0.0]	10,9827%
	[0.6 0.25 0.0]	[0.2 0.7 0.0]	[0.2317 0.6644 0.0]	6,5472%
	[0.6 0.2 0.0]	[0.2 0.7 0.0]	[0.2281 0.6648 0.0]	6,1842%
	[0.5 0.3 0.0]	[0.0 0.8 0.0]	[0.0202 0.7635 0.0]	5,2176%
	[0.0 0.8 0.0]	[0.6 0.4 0.0]	[0.5661 0.4223 0.0]	5,6285%
	[0.2 0.7 0.0]	[0.6 0.4 0.0]	[0.5652 0.4242 0.0]	5,8787%
2º Quadrante	[-0.6 0.2 0.0]	[-0.2 0.75 0.0]	[-0.2471 0.6853 0.0]	10,3076%
	[-0.6 0.2 0.0]	[0.0 0.8 0.0]	[-0.0509 0.7331 0.0]	10,5121%
	[-0.4 0.3 0.0]	[-0.2 0.75 0.0]	[-0.2025 0.7368 0.0]	1,7250%
	[-0.4 0.4 0.0]	[0.0 0.75 0.0]	[-0.0046 0.7336 0.0]	2,2734%
	[-0.4 0.4 0.0]	[0.0 0.7 0.0]	[-0.0636 0.7005 0.0]	9,0794%
	[-0.4 0.5 0.0]	[0.0 0.6 0.0]	[-0.0627 0.6064 0.0]	10,5101%
	[-0.6 0.0 0.0]	[0.0 0.7 0.0]	[-0.0633 0.7042 0.0]	9,0677%
	[0.0 0.7 0.0]	[-0.6 0.0 0.0]	[-0.5739 0.0304 0.0]	6,6808%
3º Quadrante	[-0.6 -0.2 0.0]	[-0.2 -0.75 0.0]	[-0.2660 -0.6774 0.0]	12,6437%
	[-0.6 -0.2 0.0]	[0.0 -0.8 0.0]	[-0.0353 -0.7647 0.0]	6,2392%
	[-0.4 -0.3 0.0]	[-0.2 -0.75 0.0]	[-0.2106 -0.7059 0.0]	5,8457%
	[-0.4 -0.4 0.0]	[0.0 -0.75 0.0]	[-0.0335 -0.7445 0.0]	4,5221%
	[-0.4 -0.4 0.0]	[0.0 -0.7 0.0]	[-0.0331 -0.7011 0.0]	4,7342%
	[-0.4 -0.5 0.0]	[0.0 -0.6 0.0]	[-0.0326 -0.6033 0.0]	5,4544%
	[-0.6 0.0 0.0]	[0.0 -0.7 0.0]	[-0.0336 -0.6932 0.0]	4,8940%
4º Quadrante	[0.5 0.0 0.0]	[0.0 -0.8 0.0]	[0.0050 -0.7554 0.0]	5,6126%
	[0.6 -0.2 0.0]	[0.1 -0.6 0.0]	[0.1328 -0.5986 0.0]	5,3996%
	[0.5 -0.5 0.0]	[0.0 -0.5 0.0]	[0.0011 0.4978 0.0]	0,4991%
	[0.6 -0.25 0.0]	[0.2 -0.7 0.0]	[0.2347 -0.6878 0.0]	5,0484%
	[0.6 -0.2 0.0]	[0.2 -0.7 0.0]	[0.2326 -0.6980 0.0]	4,4848%
	[0.5 -0.3 0.0]	[0.0 -0.8 0.0]	[0.0017 -0.7563 0.0]	5,4711%
	[0.0 -0.8 0.0]	[0.6 -0.4 0.0]	[0.5981 -0.4445 0.0]	6,1732%

Figura 4.4: Tabela do teste de precisão.

Assim como no teste de desempenho, foi levantado, a partir dos dados da tabela, a média e o desvio padrão de cada quadrante do círculo unitário e do total. Os resultados são apresentados na tabela abaixo.

Quadrante	**1**	**2**	**3**	**4**	**Total**
Média	7,2896%	7,5195%	6,3333%	4,6698%	**6,5165%**
Desvio padrão	2,6703%	3,6313%	2,8504%	1,9103%	**2,9349%**

Tabela 4.2: Médias e desvios padrão.

A partir desses resultados foi possível perceber que o quarto quadrante obteve o melhor resultado em relação a média e desvio padrão com um erro médio abaixo de 5% e esse resultado mais próximo do objetivo demonstra que a forma como o sistema resolve as situações para o quarto quadrante apresenta uma confiabilidade maior, que para os outros e, portanto, serve como um modelo comparativo para ajustes do próprio sistema. Outra situação possível de visualizar a partir dos dados foi que, no segundo quadrante, o sistema apresentou alguns dos melhores e piores resultados para os pontos testados demonstrando uma possível inconsistência do sistema para trabalhar nessa região com uma melhor capacidade de acerto.

Capítulo Cinco

Conclusão

Como demonstrado ao longo do trabalho existe a necessidade de que sistemas críticos possuam elementos para garantir a confiabilidade do processo em que estão inseridos, permitindo que possam entregar de forma segura e eficiente aquilo para o qual foram projetados. Entretanto, as falhas fazem parte do processo produtivo e é necessário utilizar algumas técnicas que permitam que esses sistemas que são tão sensíveis possam superar essas dificuldades e não se tornarem um problema para o proprietário ou causarem acidentes com os operadores. Nesse sentido, foram pensadas as técnicas de tolerância que tem a capacidade de aumentar a disponibilidade dos sistemas pelo fato de serem capazes de burlar a falha, como é no caso da redundância, ou detectar e reconfigurar a forma como seu funcionamento ocorre.

O objetivo principal desse trabalho foi verificar através de uma simulação numérica a viabilidade técnica de buscar uma implementação real em um sistema robótico e para isso foi escolhido um manipulador com dois eixos por se tratar da mesma configuração dos braços que ligam as hastes do robô de inspeção, objeto de estudo, e com isso verificar se as técnicas de tolerância a falha conseguiriam entregar resultados promissores em situações em que o sistema seria incapaz de continuar. Através do desenvolvimento demonstrado na metodologia foi possível obter um manipulador de dois eixos com a capacidade de verificar a presença de uma falha simulada pelo próprio sistema, retornar a um ponto que seja possível garantir a ausência de falhas e permitir o replanejamento de trajetória quando o ponto objetivo é factível, pois em situações em que o ponto final é percebido como pertencente a zona de falha é feita uma trajetória para um ponto conhecido com a menor distância possível.

Os resultados demonstraram que, para o sistema simulado, existiram quadrantes do círculo unitário que tiveram resultados consideravelmente superiores em questão de desempenho e precisão e outros que não obtiveram a repetibilidade de bons resultados necessária para garantir que os resultados ruins eram apenas *outliers*. Isso demonstra que ainda existe a necessidade de buscar e verificar se o problema para os resultados não tão satisfatórios existiu pela essência da falha a ser simulada e por isso as técnicas aplicadas obtiveram resultados inferiores ou se a simplificação do sistema real inseriu incertezas para a aplicação. Ao mesmo tempo, foi possível verificar a capacidade do sistema de entregar erros pequenos quando se trata de situações em que não são possíveis soluções para o ponto final, o que é muito positivo quando se olha para o robô de inspeção pelo fato dele possuir mais eixos de liberdade que permitiriam a compensação desses erro e um melhor ajuste para manter a estabilidade durante os deslocamentos.

5.1 Considerações finais

O trabalho desenvolvido permitiu fazer um apanhado desde o conceito dos sistemas até como é possível garantir que os sistemas entreguem aquilo que deles é esperado na presença de falhas utilizando as técnicas de tolerância a falha, que é um tema que tem se tornado cada vez mais central com o aumento da utilização e dependência de sistemas que precisam se manter funcionando independente do que aconteça, como é o caso dos sistemas de transação dos bancos, usinas de energia nuclear, aviões, etc. Através do desenvolvimento da simulação foi possível descobrir uma biblioteca vasta para o desenvolvimento de aplicações robóticas e compreender como são estruturados os modelos de manipuladores.

Essa pesquisa inicial abre um leque para implementações futuras tanto na própria simulação quanto no robô de inspeção, por demonstrar que é possível utilizar essas técnicas e ter resultados satisfatórios, entretanto é preciso refinar alguns procedimentos e revisitar a escolha das técnicas para certificar um sistema mais confiável que o desenvolvido originalmente. Como sugestões de melhorias e trabalhos futuros ficam:

- Fazer uma análise aprofundada sobre os modos e causas de falha do robô de inspeção;
- Utilizar ROS e Gazebo numa simulação mais próxima do sistema real;
- Implementar técnicas mais avançadas de tolerância a falha;
- Utilizar ferramentas de detecção de falhas utilizando inteligência artificial.

Referências

AGUERO, C. et al. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *Automation Science and Engineering, IEEE Transactions on*, v. 12, n. 2, p. 494–506, April 2015. ISSN 1545-5955. Citado na página 2.2.4.

ALHASSAN, A. B. et al. Investigation of aerodynamic stability of a lightweight dual-arm power transmission line inspection robot under the influence of wind. *Mathematical Problems in Engineering*, Hindawi, v. 2019, 2019. Citado 5 vezes nas páginas 1.2, 3.1.1, 3.1, 3.1.2, and 3.2.

AVIZIENS, A. Fault-tolerant systems. *IEEE Transactions on Computers*, IEEE Computer Society, Los Alamitos, CA, USA, v. 25, n. 12, p. 1304–1312, dec 1976. ISSN 1557-9956. Citado na página 2.3.

BERTALANFFY, L. von. *Teoria geral dos sistemas: fundamentos, desenvolvimento e aplicações*. [S.l.]: Vozes, 2008. Citado na página 2.1.

BERTOLINI, M.; BEVILACQUA, M.; MASSINI, R. Fmeca approach to product traceability in the food industry. *Food control*, Elsevier, v. 17, n. 2, p. 137–145, 2006. Citado na página 3.2.1.

DEBENEST, P. et al. Expliner-robot for inspection of transmission lines. In: IEEE. *2008 IEEE International Conference on Robotics and Automation*. [S.l.], 2008. p. 3978–3984. Citado 2 vezes nas páginas 1.2 and 1.3.

Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, v. 13, n. 2, p. 99–110, June 2006. ISSN 1558-223X. Citado na página 2.2.2.

FOGLIATO, F.; RIBEIRO, J. L. D. *Confiabilidade e manutenção industrial*. [S.l.]: Elsevier Brasil, 2009. Citado 2 vezes nas páginas 2.1.3.1 and 2.1.3.2.

GRISETTIYZ, G.; STACHNISS, C.; BURGARD, W. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In: IEEE. *Proceedings of the 2005 IEEE international conference on robotics and automation*. [S.l.], 2005. p. 2432–2437. Citado na página 2.2.2.

HANDBOOK-AIRFRAME, A. M. T. Vol. 2 (faa-h-8083-31). *US Department of Transportation, Federal Aviation Administration*, 2012. Citado 2 vezes nas páginas 1.2 and 1.2.

HESS, W. et al. Real-time loop closure in 2d lidar slam. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2016. p. 1271–1278. Citado na página 2.2.2.

HOBSBAWM, E. J. *A era das revoluções*. [S.l.]: Paz e Terra Rio de Janeiro, 2006. v. 4. Citado na página 2.2.

KOHLBRECHER, S. et al. A flexible and scalable slam system with full 3d motion estimation. In: IEEE. *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. [S.l.], 2011. Citado na página 2.2.2.

KOREN, I.; KRISHNA, C. M. *Fault-tolerant systems*. [S.l.]: Elsevier, 2010. Citado na página 2.3.1.

LAPRIE, J.-C. Dependability: Basic concepts and terminology. In: *Dependability: Basic Concepts and Terminology*. [S.l.]: Springer, 1992. p. 3–245. Citado na página 2.1.3.

LAPRIE, J.-C. Dependable computing: Concepts, limits, challenges. In: *Special issue of the 25th international symposium on fault-tolerant computing*. [S.l.: s.n.], 1995. p. 42–54. Citado na página 2.3.

LEE, P. A.; ANDERSON, T. Fault tolerance. In: _____. *Fault Tolerance: Principles and Practice*. Vienna: Springer Vienna, 1990. p. 51–77. ISBN 978-3-7091-8990-0. Disponível em: <https://doi.org/10.1007/978-3-7091-8990-0_3>. Citado 2 vezes nas páginas 2.3 and 2.3.

LEEMIS, L. M. Reliability. In: *Operations Research Applications*. [S.l.]: CRC Press, 2008. p. 77–117. Citado na página 2.1.3.1.

MATARIĆ, M. J. *Introdução à robótica*. [S.l.]: Editora Blucher, 2014. Citado 2 vezes nas páginas 1 and 2.2.

Nakju Doh; Choset, H.; Wan Kyun Chung. Accurate relative localization using odometry. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. [S.l.: s.n.], 2003. v. 2, p. 1606–1612 vol.2. ISSN 1050-4729. Citado na página 2.2.2.

OLIVEIRA, A. d. *Análise inteligente de falhas para apoiar decisões estratégicas em projetos de sistemas críticos*. Tese (Doutorado) — Universidade de São Paulo, 2009. Citado na página 2.1.1.

OLIVEIRA, D. d. P. R. d. Sistemas de informações gerenciais: estratégias, táticas, operacionais. rev. atual. *São Paulo: Altas*, 1999. Citado na página 2.1.

RANGEL, R. K.; KIENITZ, K. H.; BRANDÃO, M. P. Sistema de inspecao de linhas de transmissao de energia electrica utilizando veiculos aereos nao-tripulados. Sep, 2009. Citado na página 1.2.

RIASCOS, L. A. Fundamentos de robótica. *São Paulo: Plêiade*, 2010. Citado na página 2.6.

ROHMER S. P. N. SINGH, M. F. E. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2013. Www.coppeliarobotics.com. Citado na página 2.2.4.

SANDERS, A. *An Introduction to Unreal Engine 4*. USA: A. K. Peters, Ltd., 2016. ISBN 1498765092. Citado na página 2.2.4.

SICILIANO, B.; KHATIB, O. *Springer handbook of robotics*. [S.l.]: Springer, 2016. Citado na página 2.2.2.

SILVA, A. P. da; SANTOS, J. C. dos; KONRAD, M. R. Teoria geral dos sistemas: Diferencial organizacional que viabiliza o pleno entendimento da empresa. Citado na página 2.1.

SILVA, M.; MACHADO, J. T. Sistemas robóticos de locomoção - estado da arte. *Ingenium*, v. 2ª Série, p. 74 – 83, 10 2001. Citado na página 2.2.2.

SILVA, T. d. O. Os impactos sociais, cognitivos e afetivos sobre a geração de adolescentes conectados às tecnologias digitais. Universidade Federal da Paraíba, 2016. Citado na página 1.

SOMMERVILLE, I. Software engineering–eight edition. *Harlow: Pearson Education Limited*, 2007. Citado 2 vezes nas páginas 2.1.1 and 2.1.1.

TIAN, B. et al. Fault-tolerant control of a five-phase permanent magnet synchronous motor for industry applications. *IEEE Transactions on Industry Applications*, IEEE, v. 54, n. 4, p. 3943–3952, 2018. Citado na página 3.2.1.

VALENTI, A. W. et al. Testes de robustez em web services por meio de injeção de falhas. [sn], 2011. Citado na página 2.1.2.

WEBER, T. S. Tolerância a falhas: conceitos e exemplos. *Apostila do Programa de Pós-Graduação–Instituto de Informática-UFRGS. Porto Alegre*, p. 24, 2003. Citado 3 vezes nas páginas 1, 2.2, and 2.10.

WEBOTS. *http://www.cyberbotics.com*. Commercial Mobile Robot Simulation Software. Disponível em: <http://www.cyberbotics.com>. Citado na página 2.2.4.

ZHANG, H. et al. A novel fault tolerance parallel routing mechanism with network coding in hybrid wireless-optical broadband access network. In: IEEE. *2018 International Conference on Computing, Networking and Communications (ICNC)*. [S.l.], 2018. p. 822–826. Citado na página 3.2.1.

Estudo de implementação de modelo de tolerância a falhas em um robô para inspeção de linha de alta tensão.
Gabriel da Silva Santos

Salvador, Março de 2021.