UNIVERSIDADE CATÓLICA DO SALVADOR

Escola de Tecnologias

Curso de Engenharia de Software

SISTEMA DE INTELIGÊNCIA ARTIFICIAL PARA IDENTIFICAÇÃO DE IRREGULARIDADES EM AMBIENTES CONFINADOS UTILIZANDO ROBÔ QUADRÚPEDE

TCC BESM 06

Alécio José Santos Barreto
Caio Ricardo Lincoln Afonso
Filipe dos Santos Oliveira
Mateus Cerqueira Mota
Silenio Viana Leite Neto

Orientador: Drº GENARO FERNANDES DE CARVALHO COSTA

Salvador - BA
2025

Alécio José Santos Barreto
Caio Ricardo Lincoln Afonso
Filipe dos Santos Oliveira
Mateus Cerqueira Mota
Silenio Viana Leite Neto

SISTEMA DE INTELIGÊNCIA ARTIFICIAL PARA IDENTIFICAÇÃO DE IRREGULARIDADES EM AMBIENTES CONFINADOS UTILIZANDO ROBÔ QUADRÚPEDE

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Software.

Orientador: Prof. Dr.º Genaro Fernandes de Carvalho Costa

Salvador - BA
2025

Alécio José Santos Barreto
Caio Ricardo Lincoln Afonso
Filipe dos Santos Oliveira
Mateus Cerqueira Mota
Silenio Viana Leite Neto

SISTEMA DE INTELIGÊNCIA ARTIFICIAL PARA IDENTIFICAÇÃO DE IRREGULARIDADES EM AMBIENTES CONFINADOS UTILIZANDO ROBÔ QUADRÚPEDE

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como requisito parcial para a obtenção do título de Engenheiro de Software.

Salvador, 15 de dezembro de 2025

Banca Examinadora:

Prof. Dr.º Genaro Fernandes de Carvalho Costa
Universidade Católica do Salvador
Orientador

Prof. Me. Elton Figueiredo da Silva
Universidade Católica do Salvador

Prof.ª Me. Glaucya Carreiro Boechat
Universidade Católica do Salvador

AGRADECIMENTOS

Agradecemos à Universidade Católica do Salvador, na pessoa do nosso orientador Prof. Dr.º Genaro Fernandes de Carvalho Costa, pelo apoio e orientação durante o desenvolvimento deste trabalho.

Aos nossos familiares, pelo suporte incondicional e compreensão durante todo o curso.

Aos colegas de turma, pela troca de experiências e apoio mútuo ao longo desta jornada.

A todos os professores que contribuíram para nossa formação acadêmica e profissional.

E a todos aqueles que, direta ou indiretamente, colaboraram para a realização deste trabalho.

Resumo

O presente Trabalho de Conclusão de Curso propõe o desenvolvimento de um sistema inteligente de inspeção para espaços confinados, baseado em um robô quadrúpede Unitree Go2 Edu U2 equipado com câmera de profundidade Intel RealSense D435i e módulo de processamento NVIDIA Jetson Orin NX 16 GB. A pesquisa visa integrar robótica móvel, visão computacional e inteligência artificial (IA) para automatizar a identificação de anomalias estruturais como fissuras, trincas e corrosões em obras de engenharia e ambientes de risco. O sistema foi desenvolvido com base na documentação oficial do fabricante, envolvendo instalação física e elétrica, configuração de software via SDK e criação de uma interface de controle remoto e visualização em tempo real. As imagens capturadas são processadas localmente por redes neurais do tipo YOLOv8, treinadas com a distribuição Ultralytics e uma base de dados de imagens contendo corrosões e diferentes níveis de ferrugem. Os testes realizados em ambiente controlado demonstraram que o sistema é tecnicamente viável, apresentando média de precisão de aproximadamente 0,82 e latência operacional variando entre 120 e 180 ms, valores compatíveis com o uso em inspeções contínuas. Mesmo com a latência superior à prevista em simulação, o sistema manteve estabilidade, identificando irregularidades com consistência e sem comprometer a operação do robô.

Palavras-chave: robótica quadrúpede; visão computacional; inspeção; espaços confinados.

ABSTRACT

This undergraduate thesis presents the development of an intelligent inspection system for confined spaces, based on a Unitree Go2 Edu U2 quadruped robot equipped with an Intel RealSense D435i depth camera and an NVIDIA Jetson Orin NX 16 GB processing module. The research integrates mobile robotics, computer vision, and artificial intelligence (AI) to automatically detect structural anomalies such as cracks, fractures, and corrosion in engineering sites and hazardous environments. The system was developed based on the manufacturer's official documentation, involving physical and electrical installation, software configuration via SDK, and creation of a remote control and real-time visualization interface. Images captured by the robot are processed onboard using YOLOv8 neural networks, trained with Ultralytics distribution and an image dataset containing corrosion samples and varying degrees of rust. Tests performed in controlled environments demonstrated that the system is technically viable, presenting an average precision of approximately 0.82 and operational latency ranging between 120 and 180 ms, values compatible with continuous inspection use. Despite latency higher than initially predicted in simulation, the system maintained stability, identifying irregularities consistently without compromising robot operation.

Keywords: quadruped robotics; computer vision; inspection; confined spaces.

Lista de ilustrações

Lista de tabelas

LISTA DE SIGLAS E ABREVIATURAS

ABNT	Associação Brasileira de Normas Técnicas
AI / IA	Artificial Intelligence / Inteligência Artificial
AIR	Análise de Impacto Regulatório
API	Application Programming Interface
BLS	Bureau of Labor Statistics
COCO	Common Objects in Context
CPU	Central Processing Unit
CSI	Camera Serial Interface
CUDA	Compute Unified Device Architecture
DoF	Degrees of Freedom (Graus de Liberdade)
EPI	Equipamento de Proteção Individual
FPS	Frames Per Second (Quadros por Segundo)
FUNDACENTRO	Fundação Jorge Duprat Figueiredo
GPS	Global Positioning System
GPU	Graphics Processing Unit
HD	High Definition
HDMI	High-Definition Multimedia Interface
IA	Inteligência Artificial
IMU	Inertial Measurement Unit
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
mAP	mean Average Precision
MTP	Ministério do Trabalho e Previdência
NR-33	Norma Regulamentadora nº 33

PET	Permissão de Entrada e Trabalho
POP	Procedimento Operacional Padrão
RAM	Random Access Memory
RGB	Red, Green, Blue
RGB-D	RGB + Depth (Profundidade)
SDK	Software Development Kit
SINAN	Sistema de Informação de Agravos de Notificação
SIM	Sistema de Informações sobre Mortalidade
SLAM	Simultaneous Localization and Mapping
TCC	Trabalho de Conclusão de Curso
TOPS	Tera Operations Per Second
UCSal	Universidade Católica do Salvador
USB	Universal Serial Bus
YOLO	You Only Look Once

Sumário

1 Introdução

1.1 Contextualização e motivação

Espaços confinados em obras de engenharia, tais como galerias técnicas, poços, dutos, adutoras, reservatórios e câmaras subterrâneas, apresentam riscos significativos à integridade física de trabalhadores, incluindo deficiência de oxigênio, atmosferas tóxicas ou explosivas, visibilidade limitada, quedas e aprisionamento. A inspeção periódica desses ambientes é essencial para garantir a conformidade normativa e a segurança operacional, mas a execução manual dessas atividades expõe profissionais a condições adversas e, frequentemente, de difícil acesso.

Avanços recentes em robótica móvel, sensoriamento 3D e processamento embarcado com inteligência artificial (IA) viabilizam uma alternativa mais segura e eficiente: o emprego de robôs quadrúpedes equipados com câmeras de profundidade e unidades de computação de alto desempenho para inspeções assistidas e/ou semiautônomas. Nesse contexto, a presente pesquisa propõe um sistema integrado, combinando um robô quadrúpede com câmera de profundidade Intel RealSense e um módulo NVIDIA Jetson para processar, em campo, imagens e dados sensoriais, fornecendo diagnósticos rápidos e comparáveis ao histórico de ocorrências tecnológicas similares.

1.2 Problema de pesquisa

As inspeções em espaços confinados representam um dos cenários mais críticos de risco na engenharia civil, industrial e de saneamento. Segundo a NR-33 (Segurança e Saúde nos Trabalhos em Espaços Confinados) (Ministério do Trabalho e Emprego, 2006), do Ministério do Trabalho e Emprego, esses ambientes caracterizam-se pela ventilação insuficiente, acessos restritos e elevada probabilidade de ocorrência de atmosferas tóxicas, explosivas ou com deficiência de oxigênio. Esses fatores tornam a entrada humana extremamente perigosa e exigem procedimentos rigorosos de proteção.

Dados compilados pela Fundação Jorge Duprat Figueiredo (FUNDACENTRO) (FUNDACENTRO, 2022) apontam que, no Brasil, acidentes em espaços confinados continuam entre os mais letais do setor industrial, com dezenas de mortes anuais decorrentes de asfixia, intoxicação e aprisionamento. Relatórios internacionais reforçam essa preocupação: o U.S. Bureau of Labor Statistics registrou 1.030 mortes relacionadas a espaços confinados em um período de 10 anos (Bureau of Labor Statistics, 2021), evidenciando o risco global associado às inspeções manuais nesses ambientes.

Além dos riscos diretos à integridade física, inspeções humanas apresentam limitações operacionais significativas:

- dificuldade de acesso em dutos estreitos, galerias inundadas ou estruturas degradadas;
- variabilidade dos resultados, que dependem da experiência e do julgamento subjetivo do inspetor;
- baixa rastreabilidade, pois fotografias e anotações manuais não garantem padronização nem repetibilidade;
- tempo elevado de operação, principalmente em estruturas subterrâneas extensas.

Com o avanço da automação, soluções como drones, robôs com rodas e câmeras fixas passaram a ser utilizadas, mas apresentam limitações importantes. Drones, por exemplo, têm operação prejudicada por correntes de ar, poeira, espaços estreitos e ausência de GPS (NOGUEIRA et al., 2020). Robôs sobre rodas possuem dificuldades em desníveis, superfícies irregulares, pisos escorregadios e obstáculos comuns em tubulações (GOMES; SEO, 2022). Já sistemas fixos dependem de infraestrutura permanente e não cobrem grandes extensões.

Os robôs quadrúpedes, como o Unitree Go2, surgem como alternativa promissora devido à sua capacidade de:

- manter estabilidade em terrenos irregulares;
- transpor degraus, curvas e passagens estreitas;
- carregar sensores adicionais (LiDAR, câmeras RGB-D, módulos computacionais).

Entretanto, apesar da plataforma oferecer mobilidade avançada, não há solução nativa capaz de realizar inspeção inteligente. De fábrica, o robô fornece apenas captura de imagens e sensores básicos, sem processamento analítico ou detecção automatizada de anomalias estruturais.

Ao mesmo tempo, a identificação de fissuras, trincas e corrosões requer modelos de visão computacional robustos, como YOLO, ResNet e U-Net, técnicas que, segundo Zhou et al. (2025), atingem alta precisão quando bem treinadas, mas que dependem de processamento computacional significativo. Em ambientes confinados, a conectividade externa geralmente é limitada ou inexistente, inviabilizando o uso de computação em nuvem.

Essa restrição torna essencial o uso de processamento embarcado, como o módulo NVIDIA Jetson Orin NX, capaz de executar redes neurais diretamente no local da inspeção, com baixa latência e sem dependência externa (NVIDIA Corporation, 2023).

Assim, identifica-se uma lacuna técnica e científica: não há, atualmente, soluções consolidadas que integrem robótica quadrúpede, visão de profundidade e IA embarcada para inspeção de anomalias estruturais em espaços confinados, garantindo precisão, rastreabilidade e segurança operacional.

Diante disso, emerge a questão central deste trabalho:

> Como integrar um robô quadrúpede equipado com câmera de profundidade e processamento de IA embarcado para detectar e classificar anomalias estruturais em espaços confinados, reduzindo a exposição humana ao risco e aumentando a precisão das inspeções?

Este problema orienta a proposta do presente TCC, que busca desenvolver um sistema completo combinando hardware avançado, visão computacional e inteligência artificial embarcada.

1.3 Hipótese

A hipótese central é que a integração de um robô quadrúpede com câmera de profundidade e processamento embarcado (módulo NVIDIA Jetson) permitirá capturar, processar e analisar imagens em tempo quase real, identificando anomalias estruturais por meio de modelos de visão computacional e comparando o resultado com um banco de imagens rotuladas, fornecendo relatórios consistentes para suporte à decisão técnica e, com isso, propondo-se como uma alternativa segura para inspeções em espaços confinados, reduzindo a exposição humana a ambientes de risco.

1.4 Justificativa

A proposta tem relevância pela necessidade de mitigar riscos ocupacionais e elevar a qualidade da inspeção em locais de difícil acesso. A robótica quadrúpede oferece mobilidade superior em terrenos irregulares, degraus e passagens estreitas; a visão de profundidade auxilia na percepção de obstáculos e na documentação métrico-espacial; e o processamento embarcado viabiliza respostas rápidas *in loco*, reduzindo dependências de conectividade e latência.

Além disso, a padronização de relatórios e a comparabilidade histórica contribuem para a gestão de ativos e para estratégias de manutenção preditiva. Elementos técnicos e de integração necessários a este projeto são abordados pela documentação oficial do fabricante do robô (conceitos de SDK, integração de cargas úteis e atualização de módulos), que fundamenta a arquitetura proposta.

1.5 Objetivo geral

Desenvolver e validar a viabilidade de um sistema de inspeção para espaços confinados baseado em robô quadrúpede, visão computacional e integração da câmera de profundidade Intel RealSense e módulo NVIDIA Jetson, capaz de detectar e classificar anomalias estruturais em imagens capturadas durante missões de inspeção, comparando os achados com um banco de dados histórico treinado no modelo.

1.6 Objetivos específicos

a) Integrar hardware: montar e fixar a câmera de profundidade e o módulo de processamento no robô, respeitando padrões mecânicos, elétricos e de dados (payload/expansion), e estabelecer a comunicação com o sistema de controle.
b) Implementar software: configurar o ambiente (Jetson/SDK), realizar a comunicação com o robô e sensores, e implementar os serviços de aquisição, registro e sincronização de dados.

c) Desenvolver *front-end*: construir uma interface para teleoperação, visualização em tempo real (RGB/profundidade) e captura de evidências (fotos e metadados).
d) Preparar dataset: selecionar e organizar uma base de dados de imagens rotuladas adequada ao problema de corrosão estrutural.
e) Treinar modelo de visão computacional: treinar e otimizar modelos da família YOLO para detecção/segmentação, ajustando hiperparâmetros para execução embarcada em tempo quase real no módulo Jetson, considerando limitações reais de FPS e latência.
f) Validar: definir e executar cenários de teste em ambiente simulado e real/controlado, avaliando desempenho com métricas objetivas como precisão, F1-score, latência, taxa de vazão, degradação de desempenho e autonomia do sistema.
g) Documentar procedimentos: registrar o passo a passo de montagem, integração e atualização dos módulos, seguindo a documentação oficial do fabricante (SDK, Payload e Module Update).

1.7 Metodologia resumida

- Planejamento: seleção dos componentes (robô quadrúpede, câmera Intel RealSense D435i, Jetson Orin NX 16 GB), levantamento dos requisitos de energia, dados, fixação e rede.
- Montagem e Integração: fixação mecânica e elétrica do *payload* (câmera + Jetson) no robô, cabeamento, balanceamento e proteção; configuração de interfaces (USB/Ethernet/CSI, conforme suporte) e serviços de sistema (SDK/ROS/daemons).
- Configuração de Software: instalação do SDK do robô, bibliotecas da câmera e toolchains do Jetson; implementação de nós/serviços de controle, *telemetry* e *logging*.
- IA e Dados: *training* e *validation* dos modelos; otimização para execução embarcada.
- Atualização e Manutenção: procedimentos de *module update* (firmware/software), *rollback* e boas práticas de confiabilidade.

1.8 Escopo e limitações

O projeto foca ambientes de teste representativos de espaços confinados (escala reduzida e/ou cenários controlados) e teleoperação, não sendo objetivo nesta etapa a implementação de navegação totalmente autônoma. Restrições práticas incluem iluminação e partículas em suspensão (que podem afetar sensores), autonomia energética, largura de banda para *streaming* e qualidade do *dataset* para IA.

1.9 Contribuições esperadas

a) Arranjo integrado (hardware + software) replicável para inspeções;

b) *Pipeline* de visão com detecção/relato de anomalias e comparação histórica;
c) Interface de operação e procedimentos operacionais padrão (POP);
d) Relatórios padronizados, favorecendo rastreabilidade e tomada de decisão.

1.10 Organização do trabalho

O texto segue a estrutura acadêmica adotada: Introdução; Revisão de Literatura; Metodologia; Desenvolvimento e Resultados; Discussão; Conclusão; Referências, conforme as diretrizes institucionais previamente registradas (Universidade Católica do Salvador, 2022) e normas técnicas de documentação vigentes (ABNT, 2018).

2 Revisão de Literatura

Este capítulo apresenta a fundamentação teórica necessária para a compreensão do sistema proposto, abordando os principais conceitos, tecnologias e trabalhos relacionados que embasam esta pesquisa. Inicia-se com a caracterização dos espaços confinados e os riscos ocupacionais associados, seguida pela descrição detalhada da plataforma robótica Unitree Go2, da câmera de profundidade Intel RealSense D435i e do módulo de processamento NVIDIA Jetson Orin NX. Em seguida, são apresentados os conceitos de integração de hardware e software, os modelos de visão computacional para detecção de anomalias estruturais e uma análise comparativa dos trabalhos relacionados na literatura. Por fim, apresenta-se uma síntese dos fundamentos revisados que sustentam a proposta deste trabalho.

2.1 Espaços confinados e segurança ocupacional

De acordo com a Norma Regulamentadora nº 33 (NR-33), espaços confinados são áreas ou ambientes não projetados para ocupação humana contínua, que possuem meios limitados de entrada e saída e nos quais existe ou pode existir atmosfera perigosa, incluindo deficiência ou enriquecimento de oxigênio, presença de contaminantes tóxicos e risco de explosão.

O Guia Técnico da NR-33 complementa essa definição ao destacar que esses espaços geralmente apresentam ventilação natural insuficiente, geometrias complexas, dificuldade de movimentação interna e elevada probabilidade de aprisionamento ou asfixia, sendo encontrados em diversos setores, como saneamento, indústria química, sucroalcooleira, construção civil e energia.

Nesses ambientes, a combinação de fatores físicos, químicos e organizacionais torna o trabalho particularmente crítico. Estudos apontam que os principais riscos associados a espaços confinados incluem: atmosferas pobres em oxigênio ou enriquecidas, presença de gases inflamáveis (como metano), contaminantes tóxicos, possibilidade de alagamento, soterramento, choques elétricos, explosões, incêndios e temperaturas extremas. A NR-33, ao tratar de "atmosfera perigosa", enfatiza que qualquer condição que envolva deficiência ou enriquecimento de oxigênio, presença de contaminantes capazes de causar danos à saúde ou configuração explosiva deve ser identificada e controlada antes da entrada de trabalhadores.

A gravidade desses riscos é evidenciada pelos dados oficiais de acidentes. O Relatório de Análise de Impacto Regulatório (AIR) da NR-33, elaborado pelo Ministério do Trabalho e Previdência, mostra que entre 2011 e 2020 foram registrados, com base em códigos da CID-10 relacionados à asfixia (T71) e confinamento/aprisionamento em ambiente pobre em oxigênio (W81), 280 acidentes típicos de trabalho possivelmente associados a espaços confinados, dos quais 244 (87%) envolveram asfixia, estrangulamento ou afogamento (Ministério do Trabalho e Previdência, 2021).

O mesmo relatório indica que, ao se cruzar dados de notificação (SINAN) e mortalidade (SIM) entre 2008 e 2020, cerca de 72,7% dos acidentes de trabalho estimados em espaços confinados resultaram em óbito, demonstrando uma taxa de letalidade extremamente elevada quando comparada a outros tipos de ambiente laboral.

Em termos internacionais, dados do U.S. Bureau of Labor Statistics (BLS) indicam que, apenas entre 2011 e 2018, 1.030 trabalhadores morreram em acidentes envolvendo espaços confinados nos Estados Unidos, com destaque para silos, tanques, galerias subterrâneas e valas de escavação (U.S. Bureau of Labor Statistics, 2020). Esses números reforçam que se trata de um problema global de segurança ocupacional.

A NR-33 estabelece, portanto, não apenas uma definição jurídica para espaços confinados, mas um modelo de gestão de riscos que inclui: mapeamento e inventário dos espaços; análise preliminar de riscos; elaboração de procedimentos específicos; emissão de Permissão de Entrada e Trabalho (PET); monitoramento contínuo da atmosfera; uso de EPIs e EPCs; e capacitação obrigatória de trabalhadores autorizados, vigias e supervisores de entrada.

Mesmo com esse arcabouço normativo, o Relatório AIR da NR-33 destaca que acidentes em espaços confinados continuam ocorrendo, muitas vezes de forma "em cadeia", quando trabalhadores não treinados tentam resgatar colegas sem equipamentos e procedimentos adequados (MTP, 2021). A análise de sistemas de gestão de riscos implementados em usinas hidrelétricas demonstra a importância de procedimentos bem estruturados e treinamento contínuo para mitigação desses riscos (CARNETTI; CALVO, 2020).

Essa combinação de alta letalidade, complexidade de controle dos riscos e dificuldade de acesso físico torna os espaços confinados um campo particularmente sensível para investimentos em tecnologias de apoio à inspeção, como sensores inteligentes, monitoramento remoto e robôs móveis, que visam reduzir a necessidade de entrada direta de trabalhadores nesses ambientes.

2.2 Plataforma Principal

Os robôs quadrúpedes representam um avanço relevante em robótica móvel, pois simulam a locomoção de animais de quatro patas, permitindo maior estabilidade e capacidade de transpor obstáculos. A plataforma escolhida é o Unitree Go2, que incorpora sensores de visão, força e módulos de computação que viabilizam autonomia parcial e teleoperação precisa.

A Figura 1 apresenta o robô quadrúpede Unitree Go2 em sua configuração padrão. Observa-se a estrutura compacta do equipamento, com as quatro pernas articuladas que proporcionam mobilidade em terrenos irregulares, além dos sensores frontais integrados (câmera HD e LiDAR) que permitem percepção do ambiente. O design robusto e a disposição simétrica das pernas garantem estabilidade durante a locomoção e capacidade de manobra em espaços restritos.

Figura 1 – Robô Quadrúpede Unitree Go2

Fonte: Unitree Robotics (Unitree Robotics, 2025c)

De acordo com a documentação técnica do fabricante, o Go2 possui estrutura de alumínio e plásticos de alta resistência, pesa aproximadamente 15 kg e é capaz de carregar até 7 a 10 kg de equipamentos adicionais. Sua mobilidade é garantida por motores de torque elevado, controle dinâmico e sensores inerciais de alta frequência. O sistema de alimentação opera entre 28 V e 33,6 V, com potência de até 3000 W, e a velocidade máxima ultrapassa 3 m/s.

O modelo EDU, utilizado neste trabalho, é direcionado a aplicações de pesquisa e desenvolvimento, permitindo a instalação de câmeras, LiDARs, módulos de processamento e sensores complementares. O robô integra um LiDAR 4D L1, com varredura 360° × 90° e distância mínima de detecção de 0,05 m, além de uma câmera frontal HD 1280×720 com campo de visão de 120°. Conectividade via Wi-Fi 6, Bluetooth 5.2 e 4G eSIM com GPS amplia as possibilidades de controle remoto e sincronização de dados.

2.3 Câmera de profundidade Intel RealSense D435i

A Intel RealSense D435i é uma câmera estéreo ativa com IMU integrada (6 DoF), desenvolvida especificamente para aplicações de robótica, mapeamento 3D e visão computacional em tempo real (Intel Corporation, 2023). Seu conjunto de sensores permite captar simultaneamente profundidade, cor e movimento inercial, tornando o dispositivo adequado para navegação, inspeção e monitoramento em ambientes estruturados ou confinados.

Suas principais características técnicas incluem:

- Faixa operacional: 0,3 m a 3 m (ótimo desempenho em curta distância);
- Resolução de profundidade: até 1280×720 a 30 fps;
- Campo de visão (FOV): aproximadamente 87° (horizontal) × 58° (vertical);

- IMU integrada: acelerômetro + giroscópio (6 DoF), útil para estabilização e fusão sensorial;
- Peso: ~72 g, facilitando uso embarcado em robôs móveis;
- Interface de comunicação: USB 3.1, garantindo alta largura de banda para streaming simultâneo de profundidade e RGB;
- Tecnologia de captura: projeção ativa de infravermelho combinada com sensores estéreo, permitindo estimativa consistente de profundidade mesmo em ambientes com baixa textura.

A Figura 2 ilustra a câmera Intel RealSense D435i utilizada neste projeto. O dispositivo apresenta dimensões compactas e peso reduzido (aproximadamente 72 g), características essenciais para aplicações em robótica móvel. Na imagem, é possível identificar os dois sensores estéreo infravermelhos nas extremidades, o projetor de padrão IR no centro e a câmera RGB, que trabalham em conjunto para gerar mapas de profundidade precisos. O módulo IMU (Unidade de Medição Inercial) está integrado internamente ao dispositivo, não sendo visível externamente.

Figura 2 – Câmera Intel RealSense D435i com módulo IMU integrado

Fonte: Intel Corporation (Intel Corporation, 2023)

A presença do IMU permite sincronizar movimento e percepção espacial, facilitando a construção de mapas 3D e a estabilização da navegação do robô, que podem ser usados no futuro. A D435i integra-se facilmente a sistemas embarcados com suporte às bibliotecas *librealsense* e ROS, compatíveis com a arquitetura Jetson.

2.4 Módulo de Processamento NVIDIA Jetson Orin NX 16 GB

O NVIDIA Jetson Orin NX 16 GB é o módulo de computação embarcada responsável pelo processamento intensivo de IA no sistema de inspeção. Ele integra GPU Ampere, CPU ARM de oito núcleos e memória LPDDR5 de alta largura de banda, oferecendo desempenho suficiente para a execução do modelo YOLOv8-s em tempo quase real.

A Figura 3 apresenta o módulo NVIDIA Jetson Orin NX 16 GB montado na Expansion Dock específica para o robô Unitree Go2. Na imagem, observa-se o sistema de dissipação de calor com ventilador ativo, essencial para manter a temperatura do processador dentro dos limites

operacionais durante a execução de tarefas de inferência de IA. A placa carrier (Expansion Dock) disponibiliza as interfaces de comunicação necessárias para integração com o robô, incluindo portas Ethernet, USB e conectores de alimentação, conforme detalhado nas seções subsequentes.

Figura 3 – Módulo NVIDIA Jetson Orin NX 16 GB

Fonte: Unitree Robotics (Unitree Robotics, 2025c)

Especificações principais

- GPU: Ampere com 1024 CUDA Cores + 32 Tensor Cores;
- CPU: ARM Cortex-A78AE com 8 núcleos;
- Memória: 16 GB LPDDR5 (128 bits);
- Desempenho: até 100 TOPS (INT8 sparse);
- Energia: modo configurável entre 10 e 25 W;
- Periféricos: até 8 câmeras CSI ou 4 portas USB;
- Sistema operacional: JetPack (Ubuntu 20.04/22.04).

2.4.1 Interfaces do Jetson Orin NX e uso no projeto

O módulo é instalado no *Expansion Dock* do Go2, que expõe portas de alimentação, Ethernet, USB e interfaces industriais. A Figura 4 apresenta um diagrama detalhado das interfaces disponíveis na Expansion Dock, identificando cada conector e sua respectiva função. Na parte superior da dock, observam-se as portas Ethernet RJ-45 (duas unidades) e a porta M8 para radar. Na lateral, estão dispostas as portas USB-A 3.2 e USB-C (duas unidades, sendo uma com suporte a DisplayPort). A alimentação é fornecida através do conector XT30U-F, visível na extremidade da placa. A disposição organizada das interfaces facilita o roteamento de cabos e a conexão dos periféricos durante a montagem do sistema.

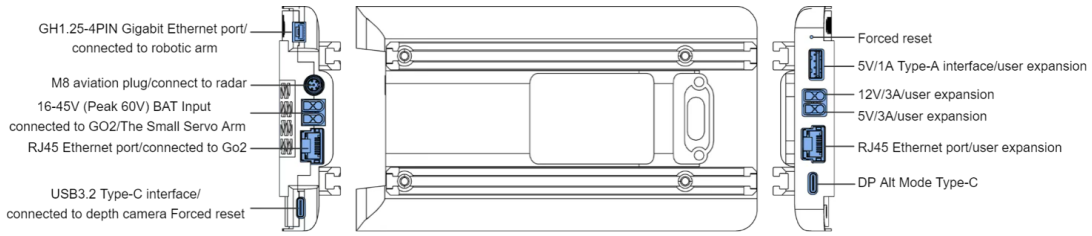

Figura 4 – Interfaces do Expansion Dock do Jetson Orin NX

Fonte: Unitree Robotics (Unitree Robotics, 2025c)

A seguir, são descritas as interfaces utilizadas no sistema e sua função operacional.

A) Alimentação: Porta XT30U-F (Power Interface)

Especificações

- Entrada: 16 a 45 V (pico 60 V)
- Saídas:
  - 12 V / 3 A (periféricos)
  - 5 V / 3 A (sensores leves)

Uso no projeto

- Alimentação principal do Jetson fornecida pelo barramento XT30 do robô.
- Saída 5 V reservada para possíveis sensores auxiliares.
- Saída 12 V disponível para expansões futuras (ex.: iluminação ou sensores externos).

B) Portas Gigabit Ethernet (RJ45: 2 portas)

Especificações

- 2 interfaces Gigabit Ethernet
- 1 porta conectada internamente ao robô
- 1 porta externa para o usuário

Uso no projeto

- **Porta interna**: comunicação Jetson ↔ controlador do Go2, garantindo baixa latência para:
  - telemetria,
  - estado dos motores,
  - IMU,
  - sincronização do pipeline de IA.
- **Porta externa**: reservada para rede local ou SSH quando necessário.

C) Porta M8: Ethernet + Alimentação (Radar Interface) *(não utilizada, mas documentada)*

Especificações

- 1 canal Gigabit Ethernet
- Saída 12 V / 2 A

Uso no projeto

- Não utilizada nesta fase, mas compatível com:

  - LiDARs,
  - radares 3D,
  - sensores industriais de alta potência.

Exemplos de aplicações futuras incluem: desenvolvimento de navegação autônoma, mapeamento 3D em tempo real e integração com sistemas de SLAM avançados.

D) Porta USB-A 3.2 Gen2 (host)

Especificações

- 10 Gbps
- 5 V / 1 A

Uso no projeto Conexão de um hub USB 3.0, permitindo:

- adaptador Wi-Fi externo (mais estável que o Wi-Fi nativo do robô),
- acesso a HDMI por meio de conversor USB-C → HDMI,
- periféricos de bancada (teclado/mouse),
- armazenamento externo para coleta de dados.

E) USB Type-C com DP Alt Mode (USB 2.0 + DisplayPort 1.4)

Especificações

- Modo USB host
- DisplayPort 1.4 (saída de vídeo)
- 5 V / 3 A

Uso no projeto

- Acesso ao ambiente gráfico Ubuntu durante configuração.
- Utilizada para depuração, instalação de dependências e ajustes de interface.

F) USB Type-C 3.2 Gen2 (host)

Especificações

- 10 Gbps
- 5 V / 2 A

Uso no projeto

- Conexão da câmera Intel RealSense D435i, que exige alta largura de banda para transmitir simultaneamente:
  - RGB,
  - Depth,
  - IMU,

  sem perda de quadros.

G) GH1.25-4PIN: Porta Ethernet Industrial

Especificações

- 1 canal Gigabit Ethernet dedicado

Uso no projeto

- Não utilizada nesta etapa.
- Reservada para possíveis módulos industriais ou integrações futuras.

2.4.2 Papel da Jetson Orin NX na arquitetura do sistema

A Jetson desempenha simultaneamente as funções de:

- Unidade central de IA embarcada, executando o modelo YOLOv8-s para detecção de anomalias;
- *Gateway* de comunicação, recebendo dados da RealSense via USB-C e telemetria via Ethernet;
- Servidor local de interface, transmitindo vídeo e *overlays* para o front-end;
- Registrador de dados, armazenando imagens, inferências e metadados para posterior análise comparativa.

Essa combinação permite operação completamente local, sem necessidade de internet ou computação em nuvem.

2.5 Integração de hardware e software do Go2

A integração entre o robô quadrúpede e seus módulos adicionais é estruturada a partir da documentação oficial da Unitree (Unitree Robotics, 2025a), composta pelos guias *SDK Concepts*, *Payload* e *Module Update*. Esses documentos definem não apenas a arquitetura de comunicação e controle, mas também os padrões de integração mecânica, elétrica e de atualização do sistema.

No nível de software, o SDK da Unitree Robotics fornece APIs em C++ e Python que permitem:

- acesso à telemetria completa (postura, motores, IMU, tensão, corrente);
- envio de comandos de movimento (velocidade, orientação, postura);
- leitura e escrita de mensagens via UDP/TCP;
- sincronização temporal entre sensores e o controlador principal.

No nível físico e elétrico, o documento *Payload* especifica:

- a pinagem e corrente suportada por cada conector (USB, Ethernet, CAN, UART, GPIO);
- limites de tensão para módulos de expansão;
- posições recomendadas para instalação de payloads sem comprometer o centro de massa;
- diretrizes de roteamento de cabos e blindagem.

Por fim, o *Module Update* padroniza o processo de atualização dos componentes internos, garantindo compatibilidade entre versões do SDK, middleware e periféricos externos.

Esses três pilares (SDK + Payload + Module Update) constituem a base da integração adotada neste projeto, assegurando conformidade técnica, estabilidade operacional e segurança durante a execução das missões de inspeção.

2.6 Modelo para detecção de anomalias estruturais

O Modelo de visão computacional baseada em inteligência artificial tem sido amplamente aplicada à inspeção civil e industrial, especialmente para identificação de fissuras, trincas e corrosões em superfícies estruturais. Entre os modelos de detecção em tempo real, o YOLOv8 Small destaca-se por sua eficiência, robustez e capacidade de operar em dispositivos de *edge computing* como o NVIDIA Jetson (Ultralytics, 2025).

O YOLOv8, utilizado neste trabalho, é uma evolução das redes YOLOv5/YOLOv7, apresentando arquitetura otimizada, cabeças de detecção mais leves e suporte nativo para quantização e exportação para TensorRT, características essenciais para execução embarcada (Ultralytics, 2025).

A literatura demonstra que, quando devidamente treinado com bancos de dados rotulados e condições realistas (iluminação variável, superfícies com ruído visual, presença de ferrugem), o YOLOv8 atinge altos índices de precisão em inspeções estruturais (ZHOU et al., 2025).

Modelos complementares como U-Net e Mask R-CNN são frequentemente utilizados para segmentação precisa, enquanto redes como ResNet e EfficientNet auxiliam na extração de características profundas. Entretanto, devido à necessidade de baixa latência e processamento em campo, este projeto adota o YOLOv8-s como solução principal, equilibrando acurácia, velocidade de inferência e compatibilidade com hardware embarcado.

A execução local na Jetson Orin NX elimina dependências de conectividade externa e reduz latência, tornando viável a análise imediata de estruturas durante a inspeção.

2.7 Trabalhos Relacionados

Esta seção apresenta uma revisão de trabalhos acadêmicos e industriais relacionados ao uso de robôs quadrúpedes, sistemas de visão computacional e inteligência artificial aplicados à inspeção de estruturas e ambientes confinados. O objetivo é posicionar a presente pesquisa no contexto do estado da arte e destacar as contribuições específicas deste trabalho.

2.7.1 Robôs Quadrúpedes para Inspeção e Navegação Autônoma

A aplicação de robôs quadrúpedes em tarefas de inspeção tem ganhado destaque devido à sua capacidade de navegação em terrenos irregulares e espaços restritos. Diversos estudos exploram plataformas como Boston Dynamics Spot, ANYmal e Unitree Go1/Go2 para inspeções industriais, de infraestrutura e ambientes perigosos.

Wilhelm (WILHELM, 2025) desenvolveu uma aplicação baseada em ROS2 que permite ao robô quadrúpede Unitree Go2 EDU realizar simultaneamente navegação autônoma, mapeamento em tempo real e reconstrução 3D de ambientes internos utilizando SLAM. O sistema integrou o Hesai XT32 LiDAR (32 canais, 3D), câmera RGB-D Intel RealSense D435i e o módulo

NVIDIA Jetson Orin NX 16 GB. Foram implementados dois algoritmos de SLAM 3D: LIO-SAM e DLIO, ambos adaptados para o Go2 e o Hesai XT32. Para navegação autônoma, utilizou-se SLAM Toolbox para mapeamento 2D e Nav2 para planejamento de trajetória. Os testes foram realizados em corredores universitários, demonstrando que o sistema é capaz de gerar nuvens de pontos densas e detalhadas em tempo real, com precisão razoável em trajetórias curtas. Entretanto, imprecisões de reconstrução e deriva posicional aumentaram em varreduras mais longas ou complexas. A execução simultânea de SLAM 2D e 3D com visualização ao vivo resultou em alta carga computacional, com risco de degradação da qualidade de saída, especialmente quando executado apenas no módulo embarcado da Jetson. A operação sem fio foi viável em redes locais de curto alcance com alta largura de banda, mas apresentou limitações sob conexões instáveis.

Enquanto Wilhelm (2025) focou na navegação autônoma completa e reconstrução 3D em tempo real com múltiplos algoritmos SLAM, o presente trabalho concentra-se especificamente na detecção de anomalias estruturais (corrosão) utilizando inteligência artificial embarcada com YOLOv8-s. Wilhelm não implementou detecção de defeitos por IA, limitando-se à captura e reconstrução geométrica do ambiente. Nossa abordagem difere ao integrar um pipeline completo de visão computacional para identificação automatizada de irregularidades, enquanto Wilhelm priorizou a qualidade do mapeamento 3D e a autonomia de navegação. Além disso, nosso sistema opera com teleoperação assistida por IA, adequando-se melhor a cenários de inspeção onde a identificação precisa de defeitos é mais crítica que a navegação totalmente autônoma.

2.7.2 Interfaces de Controle e Telemetria para Robôs Quadrúpedes

O desenvolvimento de interfaces eficientes para monitoramento e controle remoto de robôs quadrúpedes é fundamental para viabilizar operações de inspeção em ambientes complexos. Soluções tradicionais baseadas em ROSbridge e servidores MJPEG apresentam limitações de latência e complexidade de configuração.

Chowdhury (CHOWDHURY, 2025) desenvolveu uma interface web multifuncional para controlar e monitorar o robô Unitree Go2 utilizando tecnologia WebRTC. O sistema foi projetado para integrar streaming de vídeo ao vivo, painel de telemetria e comandos de controle em uma única interface baseada em navegador. A arquitetura utiliza comunicação peer-to-peer (P2P) via WebRTC para minimizar latência, com backend em Python (Flask + Cyclone) executado na Jetson Orin NX e frontend em HTML/CSS/JavaScript. O sistema publica dados de sensores (IMU, força dos pés, estado das juntas, bateria, temperatura) em tempo real através de canais de dados WebRTC. Durante os testes, a interface apresentou resposta fluida e sem interrupções, com atualização simultânea de vídeo e dados. Os comandos de movimento (D-pad e gestures) foram executados com delay mínimo. A latência de rede foi significativamente reduzida em comparação com abordagens tradicionais baseadas em ROSbridge. Limitações identificadas incluem: suporte apenas para um robô por interface, ausência de autenticação robusta, falta de

visualização do LiDAR em 3D e restrição a redes locais (LAN).

Enquanto Chowdhury (2025) focou exclusivamente na interface de controle e telemetria básica sem implementar funcionalidades de inspeção ou análise inteligente, o presente trabalho desenvolveu uma interface que integra visualização de detecções de IA em tempo real, exibindo sobreposições do modelo YOLOv8-s (*small*, 11,2 milhões de parâmetros) sobre o fluxo de vídeo para identificação de corrosões. Nossa interface, denominada CCO XD4Solutions, também incorpora controle de LiDAR, painel de métricas e módulo de áudio, mas difere ao priorizar a visualização das anomalias detectadas pela IA como elemento central da operação. Além disso, nosso sistema foi desenvolvido especificamente para o contexto de inspeção de espaços confinados com foco em segurança ocupacional, enquanto Chowdhury apresentou uma solução genérica de teleoperação.

2.7.3 Controle de Locomoção e Adaptação de Gaits com Aprendizado por Reforço

O desenvolvimento de controladores robustos para robôs quadrúpedes em ambientes dinâmicos e instáveis tem sido objeto de pesquisas recentes utilizando técnicas de aprendizado por reforço. O termo *gait* (do inglês, "marcha"ou "modo de andar") refere-se ao padrão de coordenação das pernas durante a locomoção, como *trot* (diagonal), *pace* (lateral), *bound* (traseiras juntas) e *gallop* (galope). Dois trabalhos relevantes abordam aspectos complementares dessa área: adaptação de gaits a superfícies oscilantes e simetrias para transições de gaits.

Bick (BICK, 2025) investigou a locomoção de robôs quadrúpedes sob condições de perturbação vertical do solo, utilizando o robô Unitree Go2 em uma ponte experimental (HUMVIB bridge) com frequência dominante de 2 Hz. O estudo desenvolveu 18 políticas de locomoção utilizando o algoritmo Proximal Policy Optimization (PPO) e o motor de física MuJoCo, abrangendo seis padrões de gait (default, trot, pace, bound, pronk, free) e três regimes de treinamento: superfície estática e duas configurações de ponte oscilante com diferentes estratégias de regulação de altura (relativa ao solo ou à superfície da ponte). A randomização de domínio foi aplicada para permitir transferência zero-shot para o sistema físico. Os resultados demonstraram que políticas treinadas sob condições oscilantes apresentaram estabilidade e robustez significativamente superiores durante implantação no mundo real. O trabalho introduziu métricas de avaliação focadas em controle de altura, consistência de gait e adaptabilidade, destacando a importância de incluir perturbações dinâmicas do terreno durante o treinamento para locomoção resiliente em ambientes verticalmente instáveis.

Enquanto Bick (2025) concentrou-se no desenvolvimento de controladores de locomoção robustos para terrenos instáveis usando aprendizado por reforço sem integração de sensores de visão ou IA para inspeção, o presente trabalho utiliza o Unitree Go2 primariamente como plataforma móvel para inspeção com teleoperação assistida por IA. Bick não implementou detecção de anomalias ou visão computacional, focando exclusivamente na adaptação do controle loco-

motor a oscilações verticais. Nossa abordagem difere ao priorizar a estabilidade da captura de imagens e detecção de corrosão via YOLOv8-s, enquanto Bick priorizou a robustez do controle dinâmico em si. Ambos os trabalhos utilizam a mesma plataforma robótica (Unitree Go2), mas com objetivos distintos: locomoção adaptativa versus inspeção inteligente.

Ding et al. (DING et al., 2025) apresentaram um framework unificado de aprendizado por reforço que gera gaits quadrúpedes versáteis explorando simetrias intrínsecas e relações velocidade-período de sistemas com pernas dinâmicas. Os autores propuseram uma função de recompensa guiada por simetria incorporando simetrias temporal, morfológica e de reversão temporal. Implementado no robô Unitree Go2 usando PPO e Isaac Gym, o método demonstra desempenho robusto em uma faixa de velocidades tanto em simulação quanto em testes de hardware, eliminando a necessidade de trajetórias predefinidas e permitindo transições suaves entre padrões de locomoção diversos como trotting, bounding, half-bounding e galloping. Métricas de avaliação incluíram precisão de rastreamento de velocidade, consistência de gait e custo de transporte (CoT). Resultados mostraram que políticas com reforço de simetria alcançam regulação de velocidade mais precisa, padrões de footfall mais coordenados e CoT reduzido comparado a baselines sem simetria, com taxas de sucesso de recuperação acima de 97% sem ajuste específico para plataformas distintas.

Enquanto Ding et al. (2025) focaram no desenvolvimento de um framework de controle locomotor versátil baseado em simetrias para gerar múltiplos gaits dinamicamente, o presente trabalho não explora controle avançado de locomoção, utilizando os modos de gait padrão fornecidos pelo SDK do Unitree Go2 para teleoperação. Ding et al. não implementaram funcionalidades de inspeção, visão computacional ou detecção de anomalias, concentrando-se exclusivamente na geração e transição de gaits. Nossa abordagem difere ao priorizar a plataforma robótica como meio de transporte para o sistema de inspeção com IA embarcada, enquanto Ding et al. exploraram os limites dinâmicos da locomoção quadrúpede em si. Ambos os trabalhos utilizam o Unitree Go2, mas com objetivos complementares: controle locomotor avançado versus aplicação de inspeção estrutural.

2.7.4 Navegação Autônoma e Planejamento de Cobertura

A capacidade de navegação autônoma com cobertura eficiente de áreas complexas é essencial para robôs de inspeção. Trabalhos recentes abordam estratégias de planejamento de caminho em ambientes não estruturados.

Becoy et al. (BECOY et al., 2025) propuseram um método de planejamento de cobertura de caminho para escaneamento autônomo de ambientes não estruturados utilizando o robô Unitree Go2 Edu. O método utiliza o esqueleto morfológico do mapa de navegação 2D prévio via SLAM para gerar uma sequência de pontos de interesse (POIs), que é então ordenada para criar um caminho ótimo dado a posição atual do robô. Uma máquina de estados finitos controla a operação de alto nível, alternando entre navegação em direção a um POI usando Nav2 e escaneamento

do ambiente local. O sistema foi validado em ambiente interno plano sem obstáculos em cinco ensaios, avaliando eficiência temporal e alcançabilidade. O módulo de leitura de mapa e o planejador de caminho processaram mapas de largura e altura variando entre [196,225] pixels e [185,231] pixels em 2,52 ms e 1,7 ms, respectivamente, com tempo de computação aumentando com 22,0 ns/pixel e 8,17 µs/pixel. O robô alcançou 86,5% de todos os waypoints ao longo das cinco execuções. O método proposto sofreu com drift ocorrendo no mapa de navegação 2D.

Enquanto Becoy et al. (2025) desenvolveram um sistema completo de navegação autônoma com planejamento de cobertura para exploração sistemática de ambientes, o presente trabalho utiliza teleoperação assistida por IA em vez de navegação totalmente autônoma. Becoy et al. não implementaram detecção de anomalias ou análise inteligente das áreas escaneadas, focando apenas na cobertura geométrica do espaço. Nossa abordagem difere ao priorizar a identificação precisa de corrosões via YOLOv8-s durante a teleoperação, enquanto Becoy et al. priorizaram a autonomia de navegação e cobertura completa da área. Ambos os trabalhos utilizam o Unitree Go2 Edu com SLAM e sensores de profundidade, mas com estratégias operacionais distintas: navegação autônoma planejada versus inspeção teleoperada com detecção inteligente.

2.7.5 Recuperação de Quedas e Coordenação Wheel-Leg

A resiliência operacional de robôs quadrúpedes em ambientes complexos inclui a capacidade de recuperação autônoma após quedas, especialmente para plataformas híbridas wheel-leg.

Deng et al. (DENG et al., 2025) introduziram um framework baseado em aprendizado para recuperação adaptativa de quedas em robôs quadrúpedes com rodas, que combinam agilidade de pernas com velocidade de rodas para recuperação rápida e eficiente. O método une moldagem dinâmica de recompensas baseada em episódios (Episode-based Dynamic Reward Shaping) com curriculum learning para balancear exploração de manobras diversas de recuperação e refinamento de posturas precisas. Uma arquitetura assimétrica actor-critic acelera o treinamento usando informação privilegiada de simulação, enquanto observações com ruído injetado melhoram robustez sob incerteza. O estudo demonstrou que coordenação sinérgica wheel-leg reduz consumo de torque nas juntas em 15,8% e 26,2% e melhora estabilização através de mecanismos de transferência de energia. Avaliações extensivas em duas plataformas quadrúpedes distintas (KYON e Unitree Go2-W) alcançaram taxas de sucesso de recuperação de até 99,1% e 97,8% sem ajuste específico para plataforma.

Enquanto Deng et al. (2025) concentraram-se em desenvolver controladores robustos de recuperação pós-queda usando aprendizado por reforço para robôs wheel-leg, o presente trabalho não aborda recuperação autônoma ou controle avançado de locomoção, utilizando a plataforma Unitree Go2 EDU padrão (sem rodas) com teleoperação para inspeção. Deng et al. não implementaram funcionalidades de inspeção estrutural, visão computacional ou detecção de anomalias, focando exclusivamente na resiliência locomotora. Nossa abordagem difere ao priorizar a estabilidade operacional para captura de imagens de qualidade e detecção de corrosão,

enquanto Deng et al. exploraram a robustez dinâmica de recuperação de quedas. Os trabalhos utilizam variantes diferentes do Go2 (EDU versus Go2-W) com objetivos complementares: inspeção inteligente versus recuperação robusta.

2.7.6 Detecção de Anomalias Estruturais com Inteligência Artificial

Conforme revisado na seção anterior deste capítulo, a literatura apresenta diversos trabalhos sobre detecção automática de fissuras, trincas e corrosões utilizando redes neurais profundas como YOLO, Faster R-CNN, U-Net e ResNet (ZHOU et al., 2025). Esses modelos têm sido aplicados tanto em processamento em nuvem quanto embarcado, com diferentes trade-offs entre precisão e velocidade.

No contexto específico de robôs quadrúpedes Unitree Go2 aplicados à inspeção com IA embarcada para detecção de defeitos estruturais, não foram encontrados trabalhos publicados que integrem essas três componentes simultaneamente. Os trabalhos identificados sobre o Go2 focam em navegação autônoma e mapeamento 3D (Wilhelm, 2025) ou em interfaces de controle (Chowdhury, 2025), mas não abordam a detecção inteligente de anomalias estruturais como objetivo principal.

2.7.7 Análise Comparativa

Para consolidar a análise dos trabalhos relacionados e posicionar claramente a contribuição desta pesquisa, foram elaboradas duas tabelas comparativas. A Tabela 1 apresenta uma síntese dos principais trabalhos identificados na literatura sobre o robô Unitree Go2, comparando-os com o sistema proposto em termos de plataforma, foco principal, técnicas de IA/SLAM utilizadas e aplicação-alvo. Já a Tabela 2 complementa essa análise ao detalhar aspectos de desempenho, incluindo latência, tipo de processamento e principais limitações técnicas identificadas em cada trabalho.

Essa comparação estruturada permite identificar a lacuna específica preenchida por esta pesquisa: enquanto os trabalhos anteriores focaram predominantemente em navegação autônoma, controle locomotor ou interfaces de telemetria genéricas, o presente trabalho é o primeiro a integrar especificamente inteligência artificial embarcada (YOLOv8-s) para detecção automatizada de anomalias estruturais (corrosão) no contexto de inspeção de espaços confinados utilizando a plataforma Unitree Go2.

Trabalho	Plataforma	Foco Principal	IA/SLAM	Aplicação
Chowdhury (2025)	Go2 + Jetson Orin NX	Interface WebRTC	Não utiliza	Teleoperação genérica
Wilhelm (2025)	Go2 + Hesai XT32 + D435i	Navegação autônoma + SLAM 3D	LIO-SAM, DLIO	Mapeamento 3D, sem detecção de defeitos
Bick (2025)	Go2 (HUMVIB bridge)	Locomoção em superfícies oscilantes	PPO, MuJoCo	Controle de gait adaptativo
Ding et al. (2025)	Go2 (Isaac Gym)	Gaits dinâmicos com simetria	PPO, RL	Transições de gait, sem inspeção
Becoy et al. (2025)	Go2 Edu + L1 LiDAR	Navegação autônoma + cobertura	SLAM Toolbox, Nav2	Exploração, sem IA de detecção
Deng et al. (2025)	Go2-W (wheel-leg)	Recuperação de quedas	PPO, RL	Resiliência locomotora
Presente trabalho	**Go2 + D435i + Jetson Orin NX**	**Inspeção com IA**	**YOLOv8-s**	**Detecção de corrosão (precisão 0,82)**

Tabela 1 – Comparação entre trabalhos relacionados e o sistema proposto

Trabalho	Latência	Processamento	Principais Limitações
Chowdhury (2025)	Baixa (WebRTC P2P)	Jetson (telemetria)	Sem IA, apenas controle
Wilhelm (2025)	Variável (SLAM)	Jetson + PC externo	Alta carga computacional, deriva em varreduras longas
Bick (2025)	50 Hz (controle)	Simulação (MuJoCo)	Treinamento específico para oscilações, sem inspeção
Ding et al. (2025)	50 Hz (controle)	Isaac Gym (sim)	Foco em locomoção, não em aplicação de inspeção
Becoy et al. (2025)	Planner: 1-2 ms	Jetson (SLAM/Nav2)	Drift do mapa, 86,5% alcançabilidade, sem IA
Deng et al. (2025)	100 Hz (política)	Isaac Sim (treino)	Recuperação de queda, não aplicável a inspeção
Presente trabalho	**120-180 ms**	**Jetson (embarcado)**	**Latência IA, dataset genérico, teleoperação**

Tabela 2 – Comparação de desempenho e limitações técnicas

2.7.8 Posicionamento do Presente Trabalho

Com base na análise dos trabalhos relacionados sobre o robô Unitree Go2, identifica-se uma lacuna importante: embora existam trabalhos recentes focados em navegação autônoma e SLAM 3D (Wilhelm, 2025; Becoy et al., 2025), interfaces de telemetria e controle (Chowdhury, 2025), controle avançado de locomoção (Bick, 2025; Ding et al., 2025) e recuperação de quedas (Deng et al., 2025), não foram encontrados estudos que integrem especificamente inteligência artificial embarcada para detecção automatizada de anomalias estruturais em espaços confinados utilizando esta plataforma robótica.

A principal contribuição desta pesquisa reside na integração completa de três componentes (robô quadrúpede Unitree Go2, visão de profundidade Intel RealSense D435i e IA embarcada com YOLOv8-s) aplicados especificamente ao contexto de inspeção de espaços confinados com foco em detecção de corrosão. Enquanto trabalhos anteriores exploraram esses elementos separadamente ou em combinações parciais para outros fins, a presente proposta oferece:

- pipeline de detecção de anomalias estruturais por IA embarcada, executando YOLOv8-s diretamente na Jetson Orin NX;
- arquitetura modular e replicável, documentada seguindo diretrizes oficiais do fabricante (SDK, Payload, Module Update);
- interface de operação integrada (CCO XD4Solutions) com visualização em tempo quase real das detecções de IA sobrepostas ao fluxo de vídeo;
- validação experimental com protótipo físico funcional em cenário de bancada representativo de inspeções estruturais;
- análise crítica das limitações de desempenho em tempo real e propostas concretas de evolução.

Dessa forma, o trabalho se posiciona como uma contribuição prática e científica ao campo da inspeção robótica assistida por IA, diferenciando-se dos trabalhos relacionados em aspectos fundamentais:

- Em relação a Wilhelm (2025) e Becoy et al. (2025): prioriza detecção inteligente de defeitos estruturais sobre navegação autônoma e cobertura geométrica;
- Em relação a Chowdhury (2025): incorpora análise por IA como elemento central da interface, não apenas telemetria básica;
- Em relação a Bick (2025) e Ding et al. (2025): utiliza a plataforma como meio de transporte para inspeção com IA, não como objeto de estudo de controle locomotor avançado;
- Em relação a Deng et al. (2025): foca na aplicação prática de inspeção estrutural, não em robustez de recuperação pós-queda.

O sistema oferece uma base sólida para desenvolvimentos futuros em direção a inspeções técnicas mais seguras e automatizadas em espaços confinados, preenchendo a lacuna identificada entre os trabalhos de controle locomotor/navegação e aplicações práticas de inspeção com IA embarcada.

2.8 Síntese da literatura

A revisão bibliográfica evidencia a convergência de três pilares tecnológicos essenciais para inspeções avançadas: robótica quadrúpede, visão computacional 3D e inteligência artificial embarcada. Robôs como o Unitree Go2 oferecem mobilidade elevada em ambientes restritos,

enquanto sensores RGB-D, como a Intel RealSense D435i, fornecem dados visuais e geométricos adequados para diagnósticos estruturais.

No campo da IA, modelos modernos de detecção, especialmente o YOLOv8-s (*small*), disponibilizado pela distribuição Ultralytics (Ultralytics, 2025), consolidam-se como soluções eficientes para análise de anomalias estruturais. Esses modelos combinam boa precisão com capacidade de operação em dispositivos compactos, permitindo processamento totalmente local via módulos como o NVIDIA Jetson Orin NX.

A literatura também reforça a importância do processamento embarcado para inspeções em espaços confinados, onde redes instáveis, ausência de GPS e limitações físicas inviabilizam soluções baseadas em computação na nuvem.

Assim, o sistema proposto neste trabalho alinha-se ao estado da arte ao integrar:

- mobilidade do robô quadrúpede,
- percepção RGB-D,
- detecção inteligente por meio do YOLOv8-s,

tudo operando em tempo quase real no ambiente de inspeção.

Essa combinação estabelece um modelo replicável para inspeções técnicas seguras, rápidas e com maior padronização.

3 Metodologia

Este capítulo descreve os procedimentos metodológicos adotados para o desenvolvimento do sistema de inspeção proposto. São detalhadas as etapas de planejamento, seleção de componentes, integração de hardware e software, desenvolvimento da interface de usuário, preparação do banco de dados de imagens, modelagem e treinamento do modelo de inteligência artificial, além dos procedimentos de validação experimental. A metodologia foi estruturada de forma incremental, partindo da definição dos requisitos técnicos até a validação do protótipo em ambiente controlado, garantindo a reprodutibilidade dos experimentos e a conformidade com a documentação oficial do fabricante da plataforma robótica.

3.1 Planejamento do sistema

O presente trabalho propõe o desenvolvimento de um sistema de inspeção para espaços confinados baseado em um robô quadrúpede Unitree Go2 EDU, equipado com câmera de profundidade Intel RealSense D435i e módulo de processamento embarcado NVIDIA Jetson Orin NX 16 GB.

A arquitetura resultante integra robótica móvel, visão computacional e inteligência artificial embarcada, permitindo a captura, o processamento e a análise automática de imagens estruturais em tempo quase real.

O projeto foi estruturado em cinco etapas principais:

1. **Levantamento de requisitos de hardware e software**: identificação dos componentes necessários, das interfaces de comunicação e das restrições de energia, processamento e rede.
2. **Integração física e elétrica dos módulos**: acoplamento mecânico e elétrico da câmera Intel RealSense D435i e da Jetson Orin NX ao robô Unitree Go2, respeitando as diretrizes de *Payload* do fabricante.
3. **Implementação de software e comunicação**: configuração do SDK da Unitree, dos drivers da câmera e das bibliotecas de visão computacional, bem como estabelecimento da comunicação entre robô, sensores e módulo de processamento.
4. **Treinamento e otimização dos modelos de IA**: preparação do conjunto de dados, treinamento do modelo YOLOv8-s (Ultralytics) e ajustes para execução embarcada na Jetson Orin NX.
5. **Validação experimental**: execução de ensaios em ambiente controlado, avaliando o desempenho do sistema em termos de detecção de ferrugem, latência e estabilidade operacional.

3.1.1 Requisitos de hardware

- Robô quadrúpede Unitree Go2 EDU (versão para pesquisa e desenvolvimento).
- Câmera de profundidade Intel RealSense D435i, com interface USB 3.1 e IMU integrada (6 DoF).
- Módulo de processamento NVIDIA Jetson Orin NX 16 GB, responsável pela inferência de IA embarcada.
- LiDAR 4D L1 integrado ao robô, utilizado para navegação e percepção espacial.
- Fonte de alimentação do *payload* fornecida pelo próprio Go2, na faixa de 28 V a 33,6 V, conforme especificação do fabricante.
- Módulo de comunicação com suporte a Wi-Fi 6, Bluetooth 5.2 e Ethernet Gigabit, para teleoperação e troca de dados.
- Sistema operacional embarcado JetPack 5, baseado no Ubuntu 20.04 LTS, rodando na Jetson Orin NX.

3.1.2 Requisitos de software

- SDK da Unitree Robotics, utilizado para controle de movimento, acesso à telemetria e sincronização de dados do robô.
- Biblioteca da Intel RealSense (*librealsense*) para aquisição e configuração dos fluxos RGB, profundidade e IMU da câmera D435i.
- Bibliotecas de visão computacional e IA:
  - OpenCV 4.8 (OpenCV Team, 2024) (pré-processamento de imagens e operações gráficas);
  - PyTorch 2.1 (PyTorch Team, 2024) (treinamento e execução de redes neurais);
  - Ultralytics YOLOv8 8.1 (Ultralytics, 2025) (modelo principal de detecção de anomalias estruturais).
- Sistema de interface (front-end) desenvolvido em Python + Cyclone, responsável por:
  - visualização em tempo real das imagens (RGB/profundidade);
  - teleoperação do robô;
  - exibição das detecções de IA;
  - registro e armazenamento das evidências coletadas.

3.2 Montagem e integração dos módulos

A montagem do sistema embarcado teve como objetivo acoplar o módulo de alto desempenho Jetson Orin NX 16 GB e a câmera de profundidade Intel RealSense D435i ao robô quadrúpede Unitree Go2, garantindo:

- estabilidade mecânica,
- compatibilidade elétrica com o barramento de energia do robô,

- roteamento seguro dos cabos,
- comunicação confiável entre robô, sensores e módulo de processamento.

O processo de integração seguiu as instruções oficiais *Install Expansion Dock* e *Installing a depth camera D435i* fornecidas pela Unitree (Unitree Robotics, 2025c).

3.2.1 Fixação mecânica da carga extra

A fixação mecânica da Expansion Dock da Jetson Orin NX foi realizada na porção superior traseira do Go2, na posição prevista pelo fabricante para módulos de alto desempenho. O procedimento, ilustrado nas figuras da documentação oficial, pode ser resumido em três etapas principais:

1. **Abertura da carenagem superior**
   Utilizou-se a chave Allen em "L" fornecida com o robô para remover os parafusos da proteção traseira e acessar o compartimento superior. A Figura 5 ilustra este procedimento, destacando a posição dos parafusos de fixação e a direção de remoção da tampa. Este procedimento deve ser realizado com o robô desligado e em superfície estável para evitar danos aos componentes internos.

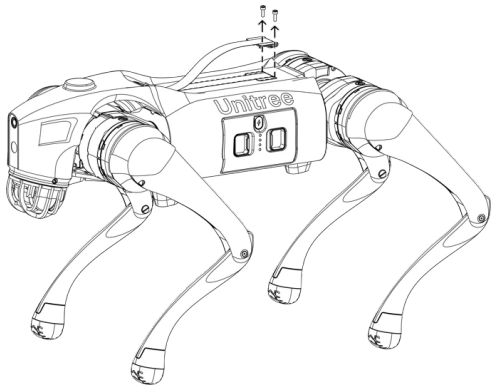

Figura 5 – Remoção da carenagem superior do Go2

Fonte: Unitree Robotics (Unitree Robotics, 2025c)

2. **Posicionamento da Expansion Dock**
   Com a tampa removida, a Expansion Dock foi alinhada aos trilhos de montagem e posicionada sobre o chassi superior do robô, respeitando o sentido indicado pelo fabricante. A Figura 6 demonstra o alinhamento correto da dock com os trilhos internos do Go2, evidenciando a direção de encaixe e os pontos de apoio que garantem a estabilidade mecânica do conjunto.

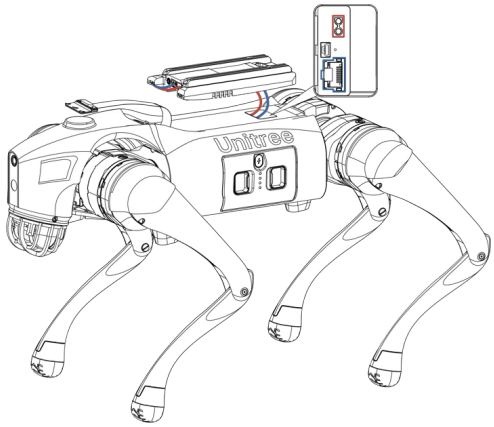

Figura 6 – Posicionamento da Expansion Dock

Fonte: Unitree Robotics (Unitree Robotics, 2025c)

Essa posição garante que os conectores de energia e Ethernet fiquem alinhados ao chicote interno do Go2.

3. **Fixação definitiva**
   A dock foi então fixada ao corpo do robô utilizando parafusos M3×50, conforme especificado pela Unitree. A Figura 7 apresenta o processo de fixação com parafusos, identificando os quatro pontos de ancoragem que distribuem uniformemente a carga mecânica e garantem a rigidez estrutural necessária para operação em movimento.

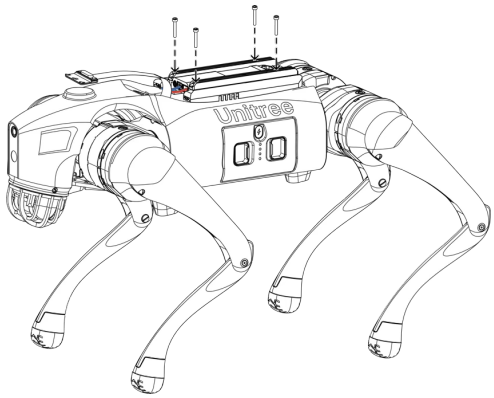

Figura 7 – Fixação definitiva da Expansion Dock

Fonte: Unitree Robotics (Unitree Robotics, 2025c)

Em seguida, a tampa original do robô foi acoplada sobre a dock, concluindo o conjunto estrutural e mantendo a proteção das conexões internas. A Figura 8 mostra o resultado da montagem com a tampa protetora reposicionada, demonstrando como a Expansion Dock se integra harmoniosamente à estrutura original do robô, mantendo a proteção contra poeira e impactos mecânicos durante a operação.

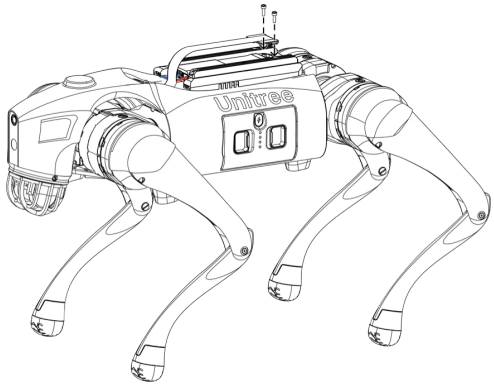

Figura 8 – Tampa acoplada sobre a Expansion Dock

Fonte: Unitree Robotics (Unitree Robotics, 2025c)

Resultado final da Jetson acoplada no equipamento:

A Figura 9 apresenta o resultado final da montagem da Jetson Orin NX no robô Go2. Na imagem, observa-se a Expansion Dock completamente integrada à estrutura do robô, com o módulo de processamento e o sistema de dissipação de calor visíveis na parte superior. A disposição do conjunto mantém o centro de massa dentro dos limites recomendados pelo fabricante, preservando a estabilidade dinâmica do robô durante a locomoção.

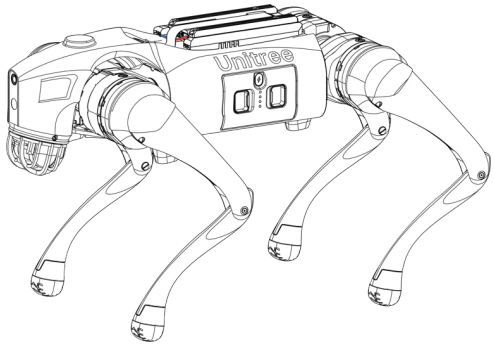

Figura 9 – Resultado final da Jetson Orin NX acoplada no Go2

Fonte: Unitree Robotics (Unitree Robotics, 2025c)

A câmera Intel RealSense D435i foi instalada na região frontal do robô, em suporte dedicado, logo acima da câmera HD original do Go2. O suporte foi projetado para fornecer um leve ângulo de inclinação para baixo, aumentando a área de observação próxima ao solo e facilitando a detecção de fissuras e irregularidades em superfícies horizontais. A Figura 10 ilustra o posicionamento da câmera RealSense, destacando sua localização estratégica acima da câmera nativa do robô e o ângulo de inclinação que permite capturar tanto superfícies verticais quanto horizontais durante a inspeção.

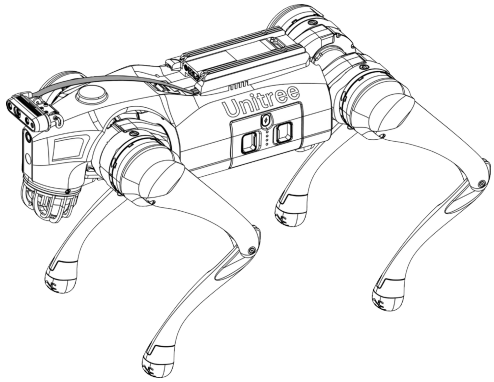

Figura 10 – Instalação da câmera Intel RealSense D435i na região frontal

Fonte: Unitree Robotics (Unitree Robotics, 2025c)

O roteamento mecânico dos cabos foi planejado para que todos os chicotes (USB, energia e Ethernet) passassem pelos canais estruturais do robô, evitando:

- pontos de torção;
- exposição excessiva;
- interferência com o movimento das pernas.

Essa organização minimiza riscos de ruptura por esforço repetitivo e reduz a chance de enrosco durante a locomoção.

3.2.2 Integração elétrica e de comunicação

A integração elétrica foi realizada com base no documento *Payload* do Go2, que especifica as faixas de tensão, correntes máximas e pinagem dos conectores disponíveis para módulos externos.

As conexões principais utilizadas no projeto foram:

- **Alimentação (XT30 / DC-IN da Expansion Dock)**
  - Entrada de 16 a 45 V fornecida diretamente pelo barramento de energia do robô, dentro da faixa de 28 a 33,6 V especificada para operação normal da bateria;
  - Essa conexão garante alimentação contínua do módulo Jetson Orin NX durante toda a missão, sem necessidade de baterias adicionais.
- **Ethernet Gigabit (RJ-45)**
  - Canal dedicado de comunicação entre a Jetson Orin NX e o controlador principal do Go2;
  - Utilizado para envio e recebimento de dados de telemetria, comandos de movimento e sincronização temporal com o pipeline de IA;
  - A escolha da Ethernet garante baixa latência e alta confiabilidade mesmo em ambientes com interferência eletromagnética.
- **USB 3.x / USB-C para a câmera Intel RealSense D435i**

  - A câmera foi ligada diretamente à porta USB-C/USB 3.1 da Jetson, assegurando a largura de banda necessária para transmissão simultânea de RGB, profundidade e dados inerciais da IMU;
  - Essa conexão permite operação contínua em tempo quase real, sem perda significativa de quadros.
- **USB-A 3.2 + Hub USB**
  - Uma porta USB-A da dock foi utilizada para conectar um hub USB, ao qual foram acoplados:
    * adaptador Wi-Fi dedicado;
    * dispositivos de armazenamento removível;
    * conversor USB-HDMI utilizado nos testes de bancada.

As interfaces adicionais GPIO, I2C, UART e SPI permaneceram disponíveis para sensores experimentais, permitindo expansão futura da plataforma sem necessidade de alterar a infraestrutura principal.

3.2.3 Configuração de software e firmware

Após a montagem física e a integração elétrica, realizou-se a configuração de software e firmware do sistema, tomando como referência o guia de inicialização rápida (*Quick Start*) do *unitree_sdk2*. O objetivo dessa etapa foi garantir que o robô Go2 e o módulo NVIDIA Jetson Orin NX operassem como um único ambiente de desenvolvimento e execução de IA, com comunicação confiável e temporização consistente.

1. Preparação e atualização do robô Go2

- Foi realizado backup da configuração original do robô (versão de firmware, parâmetros de controle e registros de log);
- Em seguida, o Go2 foi atualizado para a versão de firmware recomendada pela Unitree para uso com o módulo de alto desempenho e com o SDK utilizado neste trabalho, assegurando compatibilidade entre o controlador interno, a Expansion Dock e as interfaces de rede;
- Conforme orientação da Unitree, foram revisados os serviços de movimento (MCF/sport_mode), de modo a permitir, quando necessário, a execução de programas de controle de baixo nível sem conflito com o controlador padrão.

2. Configuração do ambiente na Jetson Orin NX

- A Jetson foi configurada com JetPack 5.1.2 (NVIDIA Corporation, 2023), baseado em Ubuntu 20.04 LTS, incluindo drivers NVIDIA, suporte a CUDA 11.4 e cuDNN 8.6, além das bibliotecas necessárias ao projeto: OpenCV 4.8 (OpenCV Team, 2024), PyTorch 2.1 (PyTorch Team, 2024) e Ultralytics YOLOv8 8.1 (Ultralytics, 2025);
- Instalaram-se as dependências indicadas para o *unitree_sdk2* (Unitree Robotics, 2025b) (cmake, gcc, build-essential, libeigen3-dev) e, em seguida, o SDK foi compilado e instalado utilizando o fluxo padrão `mkdir build`, `cmake ..`, `make install`;

- Foi habilitado o acesso remoto via SSH à Jetson, facilitando o desenvolvimento, a depuração e a atualização do código diretamente no módulo embarcado, sem necessidade de acesso físico constante.

3. Configuração de rede e integração com o SDK da Unitree

- A interface Ethernet dedicada da Jetson foi configurada na mesma sub-rede privada do computador interno do Go2, conforme recomendado pela Unitree:
  - Robô: `192.168.123.161`
  - Jetson: `192.168.123.222`
- A conectividade foi validada com comando (`ping 192.168.123.161`), verificando perda zero de pacotes e latência estável;
- Com a rede configurada, os exemplos do *unitree_sdk2* foram compilados e executados (como `go2_low_level` e `go2_stand_example`), confirmando o envio de comandos e a recepção de telemetria. Em testes de controle de baixo nível, o robô foi mantido suspenso do solo para evitar esforços indevidos nas articulações, conforme boas práticas.

4. Sincronização temporal e unificação da plataforma de inspeção

- Foi implementado um mecanismo de sincronização temporal entre o relógio da Jetson e o controlador do robô, de forma que cada frame da câmera e cada mensagem de estado recebam um carimbo temporal consistente;
- Isso permite correlacionar, com precisão, posição do robô, postura, leituras de sensores e detecções da IA.

Com essa configuração, o conjunto Go2 + Jetson Orin NX + Intel RealSense D435i passou a operar como uma plataforma integrada de inspeção: captura, processamento e funcionamento básico do sistema.

3.2.4 Desenvolvimento do software

Arquitetura do sistema

A arquitetura de software do sistema foi organizada em quatro módulos principais, executados de forma integrada sobre o módulo NVIDIA Jetson Orin NX 16 GB e acessados remotamente por meio de uma interface web. Cada módulo é responsável por uma parte bem definida da cadeia de inspeção: aquisição, controle e interação com o operador.

1. **Módulo de aquisição de dados**
   Responsável por gerenciar os sensores embarcados.
   - Controla a câmera Intel RealSense D435i, configurando resolução, taxa de quadros e streams simultâneos de RGB, profundidade e IMU (aceleração e giroscópio).
   - Gerencia o LiDAR 4D L1, quando ativado, para registro de nuvens de pontos e mapas 3D do ambiente.
   - Salva frames, mapas de profundidade e pacotes de dados sensoriais em disco, com carimbo temporal, para posterior reprocessamento e comparação histórica.

2. **Módulo de controle e telemetria**
   Implementa as rotinas de comunicação com o robô através do SDK oficial da Unitree (unitree_sdk2).
   - Envia comandos de movimentação (velocidade linear e angular, postura, início/fim de rotinas automáticas).
   - Lê continuamente telemetria: estado das juntas, orientação (yaw, pitch, roll), velocidade, temperatura média, corrente consumida e nível de bateria.
   - Publica essas informações para o módulo de interface, permitindo que o operador acompanhe, em tempo real, o estado geral do robô durante a inspeção.
3. **Módulo de processamento de IA**
   Executado na GPU da Jetson Orin NX, é o núcleo de inferência do sistema.
   - Recebe os frames RGB (e, quando necessário, profundidade) produzidos pelo módulo de aquisição.
   - Processa as imagens utilizando o modelo YOLOv8-s (Ultralytics), treinado especificamente para detecção de regiões de corrosão.
   - Associa a cada detecção: classe do defeito, confiança e coordenadas do *bounding box*.
4. **Interface gráfica (front-end web CCO XD4Solutions)**
   Desenvolvida em HTML5/CSS3/JavaScript (ES6) com backend em Python 3.10 (Flask 2.3 + Cyclone), é o painel de operação do sistema. O código HTML apresentado anteriormente corresponde a essa interface, que oferece:
   - Painel de rotinas: botões para executar scripts e ações predefinidas no robô (ex.: rotinas de caminhada, inspeção em corredor, postura de segurança).
   - Controle do LiDAR: comandos para iniciar, parar e salvar capturas de nuvens de pontos.
   - Módulo de vídeo com duas streams simultâneas (RGB e Depth), permitindo alternância da câmera principal com um botão de *picture-in-picture* (PiP).
   - Painel de métricas exibindo bateria, corrente, temperatura média, módulo da velocidade e yaw em tempo real.
   - Módulo de áudio, com upload e reprodução de sons para o robô (feedback ao operador ou interação com o ambiente).
   - Botão dedicado para lanterna/iluminação embarcada, facilitando inspeções em ambientes com baixa luminosidade.

Para complementar a descrição da arquitetura apresentada nesta seção, a Figura 11 apresenta o diagrama principal do sistema, destacando o fluxo de dados e comandos entre os componentes. O diagrama ilustra a organização em camadas do sistema: na camada física, encontram-se o robô Unitree Go2 e os sensores (câmera Intel RealSense D435i e LiDAR L1); na camada de processamento, a Jetson Orin NX executa os módulos de aquisição, controle e inferência de IA; e na camada de interface, o operador interage com o sistema por meio do painel web CCO.

As setas indicam o sentido do fluxo de dados: os sensores enviam informações para a Jetson, que processa e disponibiliza os resultados para a interface, enquanto os comandos do operador seguem o caminho inverso até o robô.

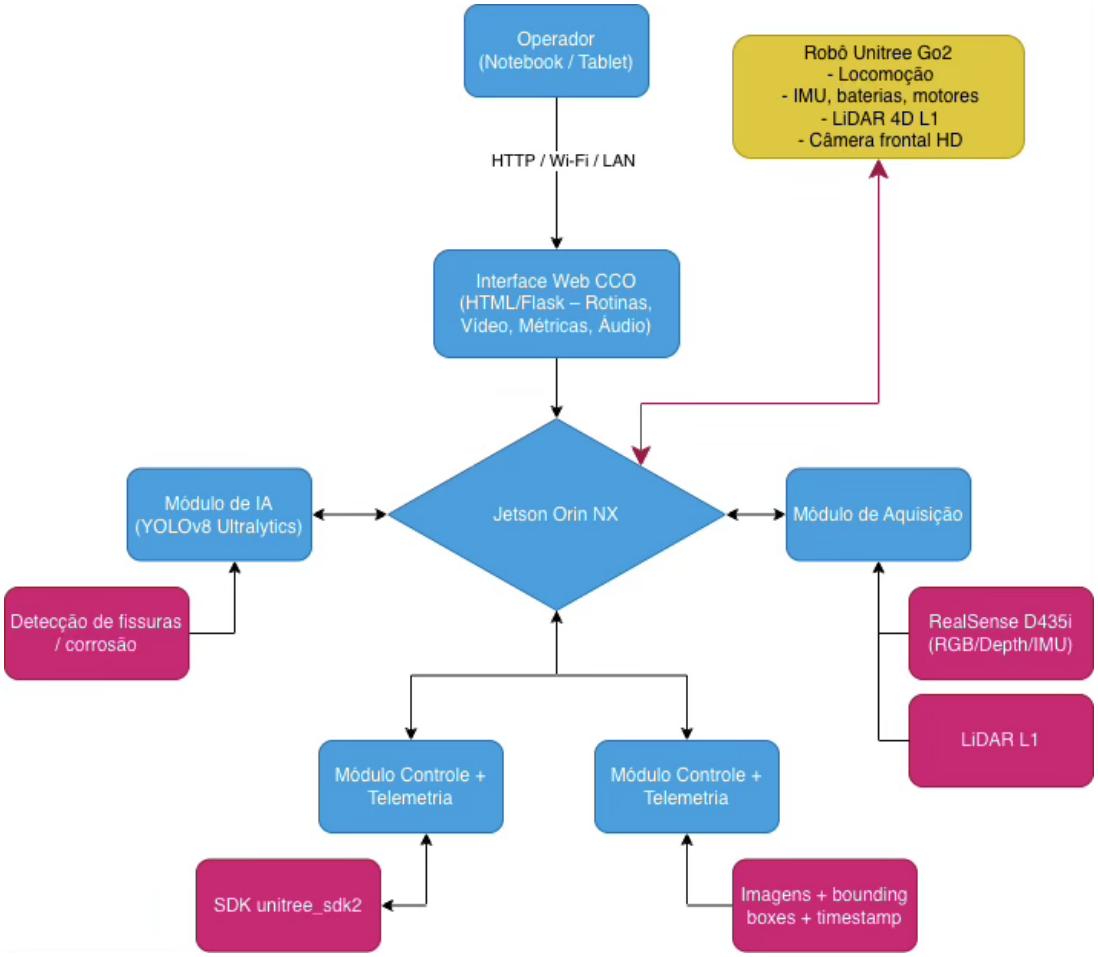

Figura 11 – Diagrama da arquitetura do sistema de inspeção

No diagrama, é possível visualizar a sequência completa de comunicação: aquisição de dados pelos sensores, envio para a Jetson, processamento pelas rotinas de IA (YOLOv8-s), armazenamento dos resultados e disponibilização das informações na interface web passada pela Jetson utilizada pelo operador. Dessa forma, o esquema resume de forma gráfica as interações entre hardware, software e inteligência artificial descritas ao longo deste capítulo.

Banco de dados de imagens

Para o treinamento do modelo de detecção de corrosão foi utilizado um conjunto de dados público, o *Rust Detection Dataset* (Roboflow Universe, 2025), disponível na plataforma Roboflow Universe (projeto `rust-detection-38s6e`, versão 1), licenciado sob CC BY 4.0. Trata-se de um dataset voltado à identificação de corrosão em superfícies metálicas em diferentes contextos (industrial, automotivo, infraestrutura), alinhado ao tipo de anomalia que se deseja detectar neste trabalho.

O conjunto é composto por 10.072 imagens anotadas, automaticamente particionadas pela plataforma em três subconjuntos:

- treinamento (train): 9.481 imagens (94%);
- validação (val): 295 imagens (3%);
- teste (test): 296 imagens (3%).

Na exportação para o formato YOLOv8 (Ultralytics), o Roboflow gerou a estrutura padrão de diretórios, apontada no arquivo `data.yaml`:

```
train: ../train/images
val: ../valid/images
test: ../test/images
nc: 10
names: ['Rust', 'car', 'copper corrosion', 'corroded-part', 'corrosion',
        'iron rust', 'mild-corrosion', 'moderate-corrosion', 'rust', 'severe-corro
```

As 10 classes definidas abrangem diferentes manifestações e intensidades de corrosão (copper corrosion, corroded-part, mild/moderate/severe-corrosion, iron rust, Rust/rust), além da classe de contexto *car*, que indica a presença de veículo na cena.

Durante o pré-processamento realizado pelo Roboflow, todas as imagens foram auto-orientadas e redimensionadas para 640×640 pixels, sem aplicação de *augmentations* (técnicas de aumento de dados que aplicam transformações como rotação, espelhamento, ajuste de brilho e contraste para expandir artificialmente o conjunto de treinamento) adicionais, ficando a cargo do Ultralytics a eventual ampliação de dados na etapa de treinamento.

As imagens e anotações são mantidas em um repositório local na Jetson Orin NX, respeitando a organização em `train/`, `valid/` e `test/`. Além desse conjunto externo, o sistema de inspeção proposto também registra, em operação, novas imagens capturadas pela câmera Intel RealSense D435i com seus respectivos metadados (data, hora, posição aproximada do robô e parâmetros de captura). Essas imagens reais de campo podem ser utilizadas futuramente para refinamento do modelo (*fine-tuning*) e para composição de um histórico de inspeções, mas não foram utilizadas no treinamento inicial descrito neste capítulo.

Modelagem de IA

A detecção automática de anomalias estruturais por corrosão foi modelada como um problema de detecção de objetos em imagens. Para isso, adotou-se como modelo base o YOLOv8-s, disponibilizado pela Ultralytics, por oferecer bom compromisso entre acurácia e velocidade, além de suporte nativo à implantação em dispositivos de borda como a NVIDIA Jetson Orin NX.

O treinamento foi realizado na forma de *fine-tuning* sobre os pesos pré-treinados em COCO, utilizando o *Rust Detection Dataset* (Roboflow Universe, 2025) descrito na Seção 3.3.2. Foram

mantidas as 10 classes originais do conjunto (diferentes tipos e severidades de corrosão, além da classe de contexto *car*), permitindo ao modelo distinguir entre níveis leves, moderados e severos de deterioração. A divisão entre treino, validação e teste seguiu exatamente o particionamento fornecido pelo Roboflow (aproximadamente 94% / 3% / 3% das imagens).

As imagens foram treinadas no formato 640×640 pixels, conforme exportação do Roboflow. Durante o treinamento, foram utilizadas as técnicas de *data augmentation* padrão do YOLOv8 (transformações geométricas e fotométricas moderadas, como *flip* horizontal, variação de escala e ajustes de cor), com o objetivo de aumentar a robustez do modelo a variações de enquadramento e iluminação. O processo de treinamento foi executado por 20 épocas, gerando ao final o *checkpoint* `best.pt`, adotado como modelo oficial deste trabalho.

Na fase de inferência, esse modelo é carregado diretamente na Jetson Orin NX, que recebe os frames RGB provenientes da câmera Intel RealSense D435i, executa o YOLOv8-s em cada imagem e retorna, para cada detecção:

- classe prevista (tipo de corrosão ou contexto);
- *bounding box* (coordenadas da região afetada);
- valor de confiança associado.

O limiar de confiança é configurável na aplicação, permitindo ajustar o comportamento do sistema para priorizar maior *recall* (mais detecções, com risco de falsos positivos) ou maior precisão (menos alertas, porém mais confiáveis), conforme o cenário de uso.

A qualidade do modelo é avaliada por meio das métricas fornecidas pelo *framework* Ultralytics (Ultralytics, 2025), incluindo precisão (P), *recall* (R), F1-score, IoU médio e mAP@0.5 / mAP@0.5:0.95, além de curvas de precisão-recall e matrizes de confusão entre classes. Os valores obtidos para essas métricas e sua interpretação serão apresentados posteriormente no capítulo de resultados.

3.3 Validação experimental

3.3.1 Ambiente de teste

Os testes práticos deste trabalho foram conduzidos em ambiente interno de laboratório, em configuração de ensaio de bancada, e não em um espaço confinado real. O foco desta etapa foi validar o funcionamento do pipeline embarcado (Go2 + Jetson Orin NX + câmera Intel RealSense D435i + modelo YOLOv8-s) em condições controladas.

O robô Unitree Go2 foi posicionado em frente a:

- chapas de aço com regiões de corrosão visível;
- peças e componentes metálicos apresentando ferrugem ou manchas com aparência semelhante à corrosão.

O ambiente possuía iluminação artificial típica de sala técnica, com pequenas variações de intensidade e sombra conforme o posicionamento do robô e das peças. A distância entre a

câmera e as superfícies metálicas foi ajustada manualmente pelo operador, de forma a representar enquadramentos próximos aos que se espera encontrar em inspeções de trechos metálicos de dutos, estruturas ou equipamentos.

Essa configuração não reproduz integralmente um espaço confinado real, mas fornece um cenário controlado suficiente para avaliar:

- a integração entre robô, câmera e módulo de IA embarcada;
- o comportamento qualitativo do modelo de detecção em superfícies corroídas;
- o impacto do processamento de IA na visualização do operador e no fluxo de operação do sistema.

Interface de usuário

A interface de usuário, denominada **CCO (Centro de Controle de Operações) XD4Solutions**, foi desenvolvida como uma aplicação web moderna utilizando tecnologias padrão da indústria. O backend foi implementado em Python 3.10, utilizando o framework Flask 2.3 para gerenciamento de rotas HTTP e a biblioteca Cyclone para comunicação assíncrona com o SDK do robô. O frontend foi construído em HTML5, CSS3 e JavaScript (ES6) puro, sem dependência de frameworks externos como React ou Vue, garantindo leveza, portabilidade e facilidade de manutenção.

A aplicação é executada diretamente na Jetson Orin NX e acessada via navegador web (Chrome, Firefox ou Safari) por qualquer dispositivo conectado à mesma rede local do robô (notebook, tablet ou smartphone). Essa arquitetura cliente-servidor elimina a necessidade de instalação de software no dispositivo do operador e permite atualizações centralizadas no servidor.

A página principal do CCO integra, em um único painel responsivo, os recursos de teleoperação, monitoramento de telemetria, visualização de vídeo com detecções de IA e controle de periféricos. O layout foi organizado em três áreas principais: (i) cabeçalho com identificação do sistema e status do script ativo; (ii) painel lateral esquerdo com controles de rotinas, LiDAR e áudio; e (iii) área principal com visualização de vídeo dual (RGB e Depth) e métricas em tempo real.

As principais funcionalidades implementadas são:

- **Conexão e controle via rede local**
  Estabelece comunicação com os serviços em execução na Jetson e com o SDK da Unitree por meio de Cyclone, permitindo o envio de comandos de movimento, execução de rotinas pré-programadas e controle do LiDAR.
- **Monitoramento de estado do robô**
  Exibe, em tempo real, métricas de telemetria como:
  - nível de bateria (%);
  - corrente consumida (A);

  - temperatura média dos atuadores (°C);
  - módulo da velocidade linear;
  - yaw (orientação) do robô.
- **Visualização de vídeo**
  Apresenta dois fluxos de vídeo simultâneos:
  - câmera de profundidade Intel RealSense D435i (RGB e Depth, com possibilidade de alternância entre visão principal e miniatura);
  - câmera nativa do Go2, utilizada como referência adicional de navegação.

  Sobre o fluxo RGB são exibidos contornos e caixas de detecção do modelo YOLOv8-s, permitindo ao operador visualizar em tempo quase real as regiões identificadas como corrosão ou anomalia.
- **Gravação de mapas 3D do LiDAR**
  Disponibiliza comandos para iniciar, interromper e salvar sessões de captura do LiDAR 4D L1, gerando arquivos que podem ser utilizados posteriormente em ferramentas de visualização e análise 3D.
- **Controle de movimentos e ações básicas**
  Inclui botões para:
  - acionar rotinas de movimento (levantar, sentar, assumir postura de inspeção);
  - ligar/desligar a lanterna ou iluminação auxiliar;
  - disparar ações específicas definidas no backend (por exemplo, scripts de teste).
- **Gravação e armazenamento local**
  Permite salvar imagens, vídeos selecionados e arquivos de log, formando um histórico de inspeções.
- **Módulo de áudio**
  Possibilita o upload e a reprodução de arquivos sonoros no robô (mensagens de voz, alertas ou sinais sonoros), contribuindo para interação com a equipe em campo e para feedback operacional durante as inspeções.

Características técnicas da interface

A interface CCO foi projetada seguindo princípios de usabilidade e experiência do usuário (UX), considerando que operadores em campo podem estar utilizando equipamentos de proteção individual (luvas, visores) que dificultam interações precisas. Por isso, os elementos interativos (botões, controles) foram dimensionados com área de toque ampliada, e o sistema de cores utiliza códigos visuais intuitivos para indicar estados: verde para operação normal, amarelo para atenção e vermelho para condições críticas.

O sistema de visualização de vídeo implementa o recurso de Picture-in-Picture (PiP), permitindo que o operador alterne dinamicamente entre visualizar o fluxo RGB ou Depth como imagem principal, enquanto o outro fluxo aparece em miniatura no canto da tela. Essa funcio-

nalidade é especialmente útil durante inspeções, pois permite focar na imagem com detecções de IA (RGB) enquanto mantém referência do mapa de profundidade, ou vice-versa.

A comunicação entre frontend e backend utiliza duas tecnologias complementares: requisições HTTP/REST para comandos de controle (iniciar/parar scripts, acionar rotinas, controlar LiDAR) e Server-Sent Events (SSE) para streaming de telemetria em tempo real (bateria, temperatura, velocidade, orientação). Essa combinação garante responsividade nos comandos e atualização contínua das métricas sem sobrecarga de rede.

A descrição detalhada da implementação da interface, incluindo trechos de código e análise da arquitetura, é apresentada na Seção 4.3.2 do Capítulo de Desenvolvimento e Resultados.

Métricas de avaliação

A avaliação do sistema proposto será realizada em dois níveis complementares: (i) desempenho do modelo de IA de detecção de corrosão e (ii) desempenho operacional do sistema embarcado no robô.

a) Métricas do modelo de IA

Para o modelo YOLOv8-s, treinado no conjunto descrito na Seção 3.3.2, serão consideradas as seguintes métricas de detecção:

- **Precisão (Precision, P)**
  Mede a proporção de detecções que são realmente corretas:

$$P = \frac{TP}{TP + FP}$$

  em que TP (true positives) representa as detecções corretas e FP (false positives) as detecções indevidas.
- **Revocação (Recall, R)**
  Mede a capacidade do modelo de encontrar os alvos existentes na cena:

$$R = \frac{TP}{TP + FN}$$

  onde FN (false negatives) são os casos em que o defeito existe, mas não foi detectado.
- **F1-Score**
  Média harmônica entre precisão e recall, utilizada como medida de equilíbrio:

$$F1 = \frac{2PR}{P + R}$$

- **mAP@0.5 e mAP@0.5:0.95**
  A *mean Average Precision* é calculada considerando múltiplos limiares de confiança e a métrica de sobreposição IoU (*Intersection over Union*) entre as caixas previstas e anotadas.

– Em **mAP@0.5**, uma detecção é considerada correta quando:

$$IoU \geq 0.5$$

– Em **mAP@0.5:0.95**, o valor é calculado como média da AP em limiares de IoU entre 0.5 e 0.95.

Além das métricas escalares, serão utilizadas matrizes de confusão e curvas precisão-recall para análise qualitativa dos erros por classe (confusão entre diferentes tipos e níveis de corrosão).

Os valores obtidos para essas métricas serão apresentados e discutidos posteriormente no capítulo de resultados, relacionando a qualidade da detecção com as restrições de processamento embarcado e de consumo de energia do robô.

Validação de robustez

Além da avaliação quantitativa do modelo em conjunto de teste, foi feita uma verificação qualitativa da robustez do sistema quanto à consistência das detecções e à estabilidade operacional do pipeline embarcado. Essa validação considerou três aspectos: (a) comportamento do modelo de IA diante de pequenas variações de cena; (b) estabilidade da execução na Jetson Orin NX; (c) sensibilidade às condições de captura adotadas nos ensaios de bancada.

A Figura 12 apresenta uma visão do ambiente de teste de bancada utilizado para validação de robustez do sistema. Na imagem, observa-se o robô Go2 posicionado diante de uma chapa metálica com regiões de corrosão visível, representando o cenário típico de inspeção. O ambiente controlado permitiu ajustar variáveis como distância da câmera, ângulo de observação e condições de iluminação, possibilitando uma análise sistemática do comportamento do modelo de IA sob diferentes condições de captura.

Figura 12 – Ensaio de validação de robustez do sistema

a) Robustez do modelo de IA

A robustez do YOLOv8-s foi analisada a partir das imagens capturadas nos ensaios de bancada com chapas e peças metálicas corroídas. Foram realizados pequenos ajustes de:

- distância entre câmera e superfície metálica;
- ângulo de observação (leve inclinação para cima/baixo e variação lateral);
- região enquadrada (aproximação e afastamento de áreas específicas de corrosão).

Em cada situação, observou-se qualitativamente se o modelo:

- mantinha detecções consistentes em áreas de corrosão mais evidentes;
- perdia detecções em regiões pequenas ou parcialmente fora de quadro;
- gerava falsos positivos em manchas não corrosivas (sujeira, tinta descascada).

Essa análise permitiu identificar que o modelo é mais robusto para regiões maiores e bem definidas de corrosão, apresentando maior variação de desempenho em defeitos discretos ou com contraste visual reduzido.

b) Robustez operacional do sistema embarcado

No nível do sistema embarcado, a robustez foi verificada monitorando o comportamento da Jetson Orin NX e da interface web durante sessões de ensaio nas quais:

- o pipeline de captura $\rightarrow$ IA $\rightarrow$ visualização permaneceu ativo por vários minutos de forma contínua;
- o robô foi ligeiramente reposicionado entre capturas, mantendo a câmera apontada para as superfícies de interesse.

Foram observados:

- gargalo de desempenho no fluxo com IA (baixa taxa de FPS e atraso perceptível), porém;
- ausência de travamentos ou necessidade de reinicialização dos serviços de captura e inferência;
- manutenção do fluxo RGB sem IA fluido, permitindo teleoperação segura mesmo quando o *overlay* do modelo apresentava atraso.

Esses ensaios indicam que, apesar da limitação de desempenho em tempo real, o sistema se manteve funcional e estável do ponto de vista operacional.

c) Sensibilidade às condições de captura

Por fim, verificou-se a sensibilidade do sistema a condições simples de captura no ambiente de laboratório, como:

- pequenas variações de iluminação artificial (mudança de posição do robô e das peças em relação às luminárias);

- alteração da distância de observação, aproximando ou afastando a câmera da chapa metálica;
- presença de manchas e marcas não corrosivas na superfície, que poderiam induzir falsos positivos.

Nessas condições, observou-se que:

- o modelo manteve desempenho razoável em distâncias curtas, com boa visibilidade da região de corrosão;
- a qualidade da detecção se degradou quando a superfície ficou muito distante ou pouco iluminada;
- manchas e marcas visuais sem corrosão efetivamente geraram detecções incorretas em alguns casos.

Embora essa validação não substitua testes em ambientes industriais reais, ela fornece uma visão inicial da resiliência do sistema frente a pequenas perturbações de enquadramento, iluminação e conteúdo visual, complementando a análise quantitativa e orientando as melhorias propostas nos capítulos seguintes.

3.3.2 Procedimentos de atualização e manutenção

A manutenção do sistema envolve tanto os componentes de software do robô Unitree Go2 quanto o ambiente de processamento na Jetson Orin NX. Para reduzir o risco de falhas durante atualizações e preservar a reprodutibilidade dos experimentos, foram adotados os seguintes procedimentos, em conformidade com o módulo Module Update e demais orientações oficiais da Unitree:

- **Versionamento e pontos de restauração do código**
  - Todo o código-fonte (scripts de controle, módulos de aquisição e interface web) foi mantido em repositório GitHub.
  - Antes de cada modificação estrutural ou atualização de dependências, era criado um commit e, quando pertinente, uma tag de versão, funcionando como ponto de restauração lógico do sistema.
- **Atualização do firmware e serviços do Go2**
  - As atualizações de firmware do robô foram executadas via ferramenta oficial Module Update, seguindo a sequência recomendada pelo fabricante (download, verificação, instalação e reinicialização).
  - Após o download, os arquivos de atualização tiveram seu checksum verificado, garantindo integridade antes da aplicação.
  - Ao final de cada ciclo de atualização, os logs de sistema foram verificados em busca de erros ou alertas.
- **Manutenção do ambiente da Jetson Orin NX**

  - Atualizações do sistema operacional e bibliotecas (JetPack, drivers NVIDIA, PyTorch, Ultralytics, librealsense) foram aplicadas de forma controlada, preferencialmente em ambiente de teste antes de serem incorporadas ao ambiente de produção.
  - Foram mantidos arquivos `requirements.txt` e registros das versões utilizadas, permitindo reconstruir o ambiente em caso de falha ou migração para outro hardware.
  - O modelo de IA final (`best.pt`) e os arquivos de configuração (`data.yaml`, parâmetros de inferência) foram armazenados em diretórios de backup e também versionados no repositório.
- **Testes pós-atualização**
  - Após cada atualização (do robô ou da Jetson), era executado um checklist mínimo:
    * teste de comunicação Ethernet (ping entre Jetson e Go2);
    * execução de exemplos do *unitree_sdk2* para validar controle e telemetria;
    * verificação da captura da câmera RealSense e do LiDAR;
    * teste rápido de inferência do modelo YOLOv8-s e exibição na interface web.

Esse conjunto de procedimentos buscou garantir que cada alteração de firmware, biblioteca ou código fosse rastreável, reversível e devidamente validada antes do uso em ensaios de inspeção.

3.3.3 Síntese da metodologia

A metodologia adotada neste trabalho combinou integração de hardware avançado com modelagem de IA embarcada, estruturada em etapas sequenciais e iterativas:

1. Planejamento do sistema, com levantamento de requisitos de hardware, software e rede, definição dos componentes (Unitree Go2, Jetson Orin NX, Intel RealSense D435i, LiDAR L1) e estudo das limitações de energia e processamento.
2. Montagem e integração física/eletrônica, seguindo a documentação oficial da Unitree para instalação da Expansion Dock, da câmera de profundidade e conexão ao barramento de energia e comunicação do robô.
3. Configuração de software e comunicação, incluindo instalação do JetPack, drivers, bibliotecas de visão computacional, SDK *unitree_sdk2* e implementação da interface web para controle e monitoramento.
4. Modelagem de IA, com seleção do YOLOv8-s, preparação do banco de imagens de corrosão, *fine-tuning* do modelo e implantação na Jetson Orin NX para inferência em tempo quase real.
5. Ensaios e validação em ambiente simulado, utilizando cenários controlados, para medir desempenho do modelo, latência de processamento, estabilidade térmica e autonomia energética do sistema.

Ao seguir as instruções de montagem e atualização do fabricante e registrar de forma sistemática cada etapa de configuração e teste, a metodologia buscou garantir compatibilidade

mecânica e eletrônica, reprodutibilidade dos experimentos e segurança operacional do robô durante as inspeções. Isso cria uma base sólida para, em trabalhos futuros, transpor o sistema para ambientes industriais reais e cenários de inspeção mais complexos.

4 Desenvolvimento e Resultados

Este capítulo apresenta a implementação prática do sistema de inspeção e os resultados experimentais obtidos durante a pesquisa. São documentados o protótipo físico final, incluindo a integração do robô Unitree Go2 com os módulos de sensoriamento e processamento, a implementação do software embarcado na NVIDIA Jetson Orin NX, o desenvolvimento da interface de usuário CCO (Centro de Controle de Operações), os resultados do treinamento do modelo YOLOv8-s e os ensaios realizados em ambiente de bancada. Os resultados são apresentados de forma quantitativa, por meio de métricas de desempenho do modelo de IA, e qualitativa, por meio da análise do comportamento do sistema durante os testes práticos. A Figura 13 apresenta uma visão geral do sistema completo de inspeção em operação durante os ensaios experimentais, demonstrando a integração entre o robô quadrúpede, a câmera de profundidade e o módulo de processamento embarcado.

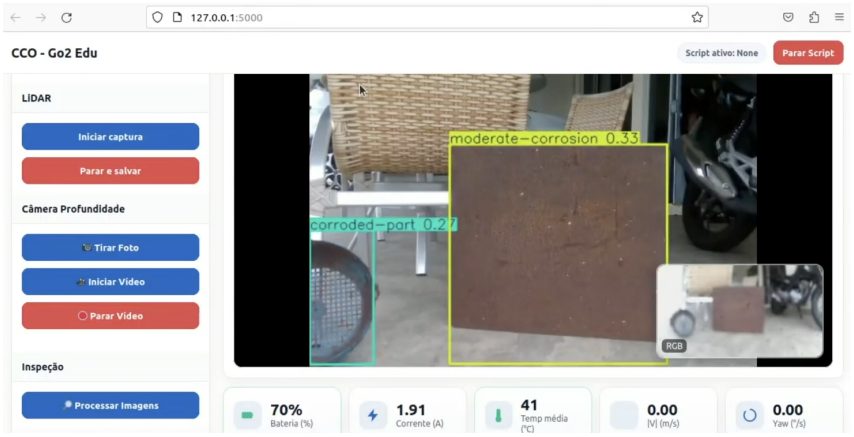

Figura 13 – Sistema completo de inspeção em operação

4.1 Visão geral da implementação

Este capítulo documenta a implementação prática do sistema de inspeção proposto, incluindo o protótipo físico montado no robô Unitree Go2, o ambiente de software embarcado na NVIDIA Jetson Orin NX, o pipeline de inteligência artificial baseado em YOLOv8-s e a interface de usuário utilizada para operação em campo.

São apresentados também os resultados experimentais obtidos em ambiente controlado, configurado para simular condições de espaços confinados (corredores estreitos, paredes próximas, trechos com baixa iluminação). As decisões de projeto e os procedimentos de implementação seguiram a documentação oficial da Unitree (SDK, Payload, Module Update), garantindo compatibilidade mecânica e eletrônica com a plataforma robótica.

4.2 Protótipo físico final

4.2.1 Robô Go2 instrumentado

A Figura 14 apresenta o protótipo físico final do robô quadrúpede Unitree Go2 EDU com o conjunto completo de sensores e processamento embarcado instalados. Na imagem, é possível identificar os principais componentes do sistema: (i) na parte superior traseira, a Expansion Dock com o módulo NVIDIA Jetson Orin NX 16 GB e seu sistema de dissipação de calor; (ii) na região frontal, a câmera de profundidade Intel RealSense D435i montada em suporte dedicado, posicionada acima da câmera HD nativa do robô; (iii) o roteamento de cabos organizado ao longo da estrutura do robô, passando pelos canais internos para evitar interferência com o movimento das pernas; e (iv) as quatro pernas articuladas em posição de repouso, demonstrando a postura padrão do equipamento.

Figura 14 – Protótipo físico do robô Go2 instrumentado

A montagem mecânica e elétrica foi realizada conforme descrito no Capítulo 3, utilizando o suporte oficial fornecido pela Unitree para a D435i e o kit de instalação da Expansion Dock. O roteamento de cabos respeitou os envelopes de movimento das pernas e o raio mínimo de curvatura dos cabos de dados, evitando interferências durante a locomoção.

Do ponto de vista dinâmico, observou-se que a adição do payload não comprometeu a estabilidade do robô em marcha lenta e em trajetórias retilíneas curtas. A postura padrão de

inspeção pode ser mantida sem ajustes significativos nos parâmetros de controle, o que indica que o centro de massa permaneceu dentro da faixa recomendada pelo fabricante.

4.2.2 Comportamento térmico e estrutural em operação

Durante os ensaios prolongados, o sistema foi submetido a ciclos de operação contínua com o modelo de IA ativo. Foram monitoradas:

- temperatura média dos atuadores do robô;
- temperatura do módulo de processamento;
- ocorrência de vibrações excessivas ou folgas mecânicas na região da dock e do suporte da câmera.

A Figura 15 ilustra uma sessão de monitoramento térmico durante operação contínua do sistema. Na imagem, observa-se o robô Go2 em ambiente de teste, com o operador acompanhando as métricas de telemetria através da interface CCO. O painel de métricas da interface exibe em tempo real os valores de temperatura dos atuadores, consumo de corrente e nível de bateria, permitindo identificar rapidamente condições que possam indicar sobrecarga térmica ou elétrica.

	Body Motor	Thigh Motor	Shank Motor
FL	27℃	27℃	29℃
FR	27℃	27℃	28℃
RL	59℃	31℃	36℃
RR	51℃	35℃	35℃

Figura 15 – Monitoramento térmico durante operação contínua

Os testes indicaram que, nas condições de carga utilizadas, o sistema manteve-se dentro dos limites térmicos recomendados, sem acionamento de mecanismos de proteção ou redução automática de desempenho. Não foram identificados afrouxos ou deslocamentos perceptíveis dos suportes após os ensaios, o que sugere que a solução de fixação adotada é adequada para missões de curta e média duração.

4.3 Implementação de software embarcado

4.3.1 Serviços na Jetson Orin NX e comunicação com o robô

A Jetson Orin NX foi configurada com JetPack (Ubuntu LTS, drivers NVIDIA e suporte a CUDA/cuDNN), além das bibliotecas necessárias ao projeto: OpenCV, PyTorch, Ultralytics YOLOv8 e o SDK *unitree_sdk2*.

Na Jetson são executados, de forma contínua, os seguintes serviços principais:

- **Serviço de aquisição**: responsável por receber os fluxos RGB/Depth/IMU da câmera Intel RealSense D435i e, quando ativado, as nuvens de pontos do LiDAR L1.
- **Serviço de IA**: carrega o modelo YOLOv8-s treinado e realiza a inferência sobre os frames recebidos, produzindo detecções (classe, bounding box, confiança).
- **Serviço de controle e telemetria**: baseado no *unitree_sdk2*, envia comandos de movimento ao robô e coleta dados como bateria, corrente, temperatura média e orientação (yaw).
- **Servidor web da interface de usuário**: implementado em Python (Flask + Cyclone), disponibiliza o painel de controle na rede local.

A comunicação entre a Jetson e o computador interno do Go2 é feita por Ethernet dedicada, em sub-rede específica, enquanto o acesso do operador ao sistema ocorre via rede local (Wi-Fi ou cabo) utilizando o endereço IP da Jetson.

4.3.2 Interface de usuário: painel CCO

A interface web desenvolvida para operação do robô, denominada **CCO (Go2 EDU)**, foi construída utilizando tecnologias web modernas e arquitetura cliente-servidor, permitindo acesso remoto via navegador sem necessidade de instalação de software adicional. As Figuras 16 e 17 apresentam a interface em dois estados distintos: aguardando conexão e em operação completa durante os testes de bancada.

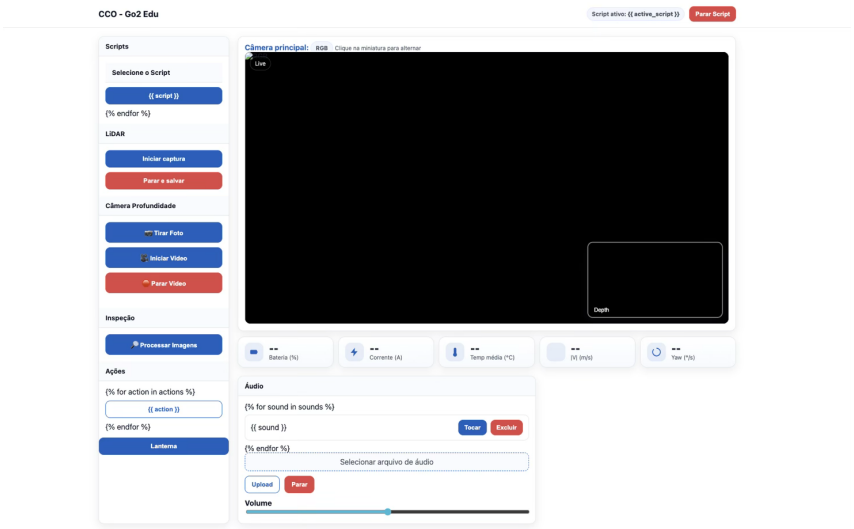

Figura 16 – Interface web CCO (Go2 EDU) em estado inicial (câmeras desconectadas)

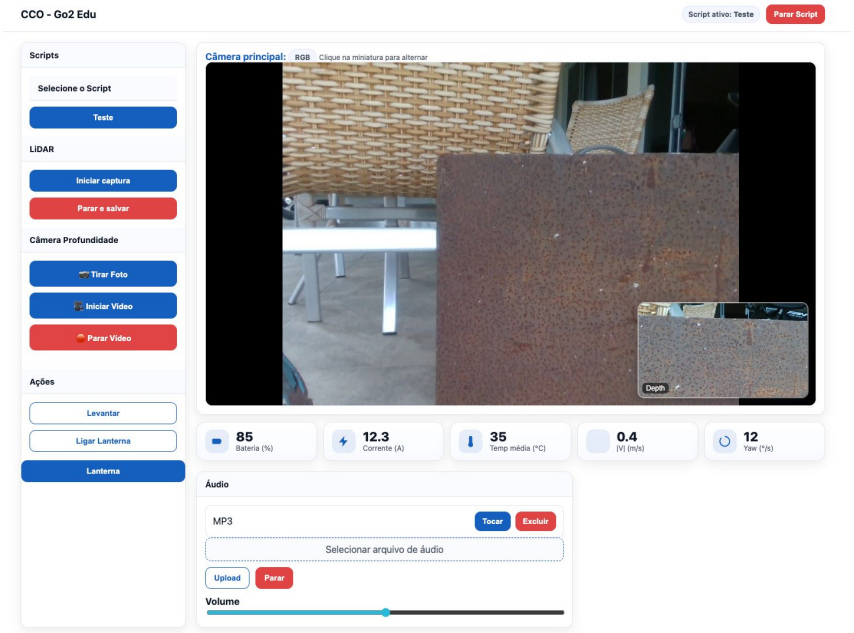

Figura 17 – Interface web CCO (Go2 EDU) em operação real com vídeo ativo e detecções de IA

A Figura 16 ilustra o estado inicial da interface, antes do estabelecimento da conexão com os streams de vídeo, mostrando a estrutura completa do layout, os controles disponíveis e o painel de métricas aguardando dados. Já a Figura 17 apresenta o sistema em pleno funcionamento, com os fluxos de vídeo RGB e Depth ativos, métricas de telemetria atualizadas em tempo real e a visualização de uma chapa metálica com corrosão durante os ensaios de bancada.

Essa comparação visual entre os dois estados é relevante para demonstrar a robustez da interface: mesmo quando os serviços de vídeo não estão disponíveis (por desconexão temporária, reinicialização do sistema ou falha de rede), a interface permanece funcional e responsiva, permitindo que o operador execute ações de controle, configure parâmetros e monitore o histórico de eventos. Essa característica é particularmente importante em cenários de campo, onde a estabilidade da rede pode variar e a capacidade de retomar operações rapidamente após reconexões é crítica para a eficiência da inspeção.

Arquitetura da interface web

A interface foi desenvolvida seguindo o padrão arquitetural cliente-servidor com comunicação assíncrona, composta por três camadas principais:

- **Frontend (Cliente)**: HTML5 + CSS3 + JavaScript vanilla, executado no navegador do operador
- **Backend (Servidor)**: Python Flask executando na Jetson Orin NX
- **Comunicação**: HTTP/REST para comandos e Server-Sent Events (SSE) para telemetria em tempo real

Estrutura HTML e layout responsivo

O layout da interface foi organizado em três áreas principais: cabeçalho (header), painel lateral esquerdo (sidebar) e área principal (main). O código HTML utiliza estrutura semântica e classes CSS modulares:

```
<header>
  <div class="header-inner">
    <div class="brand">
      <h1>CCO - Go2 Edu</h1>  <!-- Titulo principal -->
    </div>
    <div>
      <span class="chip">Script ativo: <strong id="active-script">Teste</strong></span>
      <button class="btn small danger" onclick="stopScript()">Parar Script</button>
    </div>
  </div>
</header>
<div class="page">
  <aside class="panel"><!-- Controles: Scripts, LiDAR, Camera --></aside>
  <main class="main"><!-- Area de video e metricas --></main>
</div>
```

O design responsivo adapta-se automaticamente a diferentes tamanhos de tela utilizando CSS Grid e Media Queries:

```
.page {
  display: grid;                          /* Layout em grid */
  grid-template-columns: 280px 1fr;       /* Sidebar + conteudo */
  gap: 18px;
}
@media (max-width: 1100px) {              /* Responsivo */
  .page { grid-template-columns: 1fr; }
}
```

Sistema de câmeras com Picture-in-Picture (PiP)

Um dos recursos mais relevantes da interface é o sistema de visualização dual de câmeras com troca dinâmica entre visualização principal e miniatura. A implementação utiliza manipulação direta do DOM sem recarregamento de elementos:

```
document.addEventListener('DOMContentLoaded', () => {
  const mainSlot = document.getElementById('main-slot');  // Slot principal
  const pipSlot  = document.getElementById('pip-slot');   // Slot PiP
  const rgb      = document.getElementById('cam-rgb');
  const depth    = document.getElementById('cam-depth');

  mainSlot.appendChild(rgb);    // Estado inicial: RGB principal
  pipSlot.appendChild(depth);   // Depth em miniatura

  pipBtn.addEventListener('click', () => {  // ALTERNAR CAMERAS
    const mainIsRGB = mainSlot.contains(rgb);
    if(mainIsRGB) { pipSlot.appendChild(rgb); mainSlot.appendChild(depth); }
    else { pipSlot.appendChild(depth); mainSlot.appendChild(rgb); }
  });
```

```
});
```

Essa abordagem evita reconexões de stream e garante transições suaves entre as visualizações RGB e de profundidade.

Atualização de métricas via Server-Sent Events (SSE)

A telemetria do robô (bateria, corrente, temperatura, velocidade, yaw) é transmitida em tempo real utilizando Server-Sent Events, uma tecnologia nativa dos navegadores que permite comunicação unidirecional servidor→cliente com baixa latência:

```
const source = new EventSource('/data');  // CONEXAO SSE

source.onmessage = (event) => {
  let data = JSON.parse(event.data);

  // BATERIA - Atualiza valor e estado visual
  const b = data.find(x => x.name.includes('battery'));
  if(b) {
    const v = Math.round(Number(b.value));
    document.getElementById('m_batt').textContent = v + '%';
    let state = (v <= 15) ? 'crit' : (v <= 30) ? 'warn' : 'ok';
    document.getElementById('card_batt').className = 'metric card ' + state;
  }

  // TEMPERATURA - Atualiza valor e estado visual
  const t = data.find(x => x.name.includes('temp'));
  if(t) {
    const val = parseFloat(t.value);
    document.getElementById('m_temp').textContent = val.toFixed(0);
    let state = (val >= 80) ? 'crit' : (val >= 65) ? 'warn' : 'ok';
    document.getElementById('card_temp').className = 'metric card ' + state;
  }
};
```

O sistema de estados visuais (`ok`, `warn`, `crit`) altera dinamicamente as cores dos cards de métrica, permitindo identificação rápida de condições críticas (bateria baixa, temperatura elevada).

Controles de script e ações do robô

A execução de scripts pré-programados (rotinas de movimento, inspeção) e ações individuais (levantar, ligar lanterna, controlar LiDAR) é feita por requisições HTTP POST assíncronas:

```
function runScript(name) {  // EXECUTA SCRIPT NO ROBO
  fetch('/run/' + encodeURIComponent(name), {method: 'POST'})
    .then(r => r.json())
    .then(j => {
      if(j.ok) document.getElementById('active-script').textContent = j.active;
    });
}
```

```
function lidarStart() {  // INICIA CAPTURA LIDAR
  fetch('/lidar_start', {method: 'POST'})
    .then(r => r.json())
    .then(res => { if(res.ok) alert("Captura iniciada!"); });
}
```

Essas funções comunicam-se diretamente com o backend Flask rodando na Jetson, que por sua vez interage com o SDK do robô para executar os comandos solicitados.

Sistema de áudio e upload dinâmico

A interface permite fazer upload de arquivos de áudio (.mp3, .wav, .ogg) diretamente pelo navegador, sem recarregar a página, utilizando FormData e fetch API:

```
document.getElementById('upload-form').addEventListener('submit', async (e) => {
  e.preventDefault();
  const fd = new FormData(e.target);  // DADOS DO FORMULARIO
  const r = await fetch('/upload', {method: 'POST', body: fd});  // UPLOAD
  const j = await r.json();

  if(j.ok) {  // ADICIONA ARQUIVO A LISTA DINAMICAMENTE
    const div = document.createElement('div');
    div.className = 'audio-item';
    div.innerHTML = '<span>' + j.file + '</span>' +
      '<button onclick="playSound()">Tocar</button>' +
      '<button onclick="deleteSound()">Excluir</button>';
    document.getElementById('audio-list').prepend(div);
  }
});
```

O controle de volume é implementado com um slider HTML5 range, que envia o valor ao backend apenas ao liberar o botão do mouse, evitando requisições excessivas:

```
function sliderReleased() {  // CONTROLE DE VOLUME
  const v = +document.getElementById('audio-slider').value;
  fetch('/change_volume', {
    method: 'POST',
    headers: {'Content-Type': 'application/json'},
    body: JSON.stringify({ volume: v })  // ENVIA VALOR AO BACKEND
  });
}
```

Estilização visual e identidade do sistema

A interface adota um sistema de design baseado em variáveis CSS (custom properties), garantindo consistência visual e facilitando manutenção:

```
:root {  /* VARIAVEIS CSS - DESIGN SYSTEM */
  --primary: #155FBF;   /* Cor principal */
  --accent: #2BB9D9;    /* Cor de destaque */
  --ok: #10B981;        /* Estado OK (verde) */
  --warn: #F59E0B;      /* Estado alerta (amarelo) */
  --crit: #EF4444;      /* Estado critico (vermelho) */
```

```
  --radius: 12px;
  --shadow: 0 6px 18px rgba(17,24,39,0.08);
}
```

Os cards de métricas utilizam gradientes e sombras sutis para criar hierarquia visual, com ícones SVG escaláveis e estados dinâmicos de cor:

```
.metric.card {  /* CARD DE METRICA */
  display: flex; align-items: center; gap: 12px;
  background: linear-gradient(180deg, #fff, #F8FAFF);
  border-radius: 14px; padding: 12px 14px;
  box-shadow: var(--shadow);
}
.metric.ok   { border-color: rgba(16,185,129,0.35); }   /* VERDE */
.metric.warn { border-color: rgba(245,158,11,0.45); }   /* AMARELO */
.metric.crit { border-color: rgba(239,68,68,0.5); }     /* VERMELHO */
```

Componentes visuais da interface

Conforme ilustrado na Figura 17, a interface integra os seguintes componentes principais:

- **Área de vídeo principal**, com dois fluxos simultâneos (RGB e Depth) da câmera RealSense, permitindo alternância entre qual deles aparece como principal e qual fica em modo picture-in-picture. Sobre o fluxo RGB são desenhadas, em tempo quase real, as detecções do modelo YOLOv8-s (caixas e rótulos).
- **Painel de métricas**, exibindo em tempo real:
  - nível de bateria;
  - corrente consumida;
  - temperatura média;
  - módulo da velocidade;
  - yaw do robô.
- **Seção de rotinas e ações**, com botões para chamar scripts pré-definidos (postura de inspeção, caminhar à frente, girar, entre outros), acionar o LiDAR, iniciar/parar gravações e ligar/desligar a lanterna.
- **Módulo de áudio**, permitindo upload de arquivos sonoros e reprodução diretamente no robô, além de ajuste de volume.

Nos testes realizados, a interface mostrou-se responsiva, com atualização fluida das métricas e do vídeo. A sobreposição das detecções do YOLOv8-s permitiu ao operador identificar visualmente as regiões da cena marcadas como corrosão ou anomalia, facilitando a análise em tempo quase real.

Stack tecnológico e justificativa das escolhas

A escolha das tecnologias para construção da interface web foi orientada por critérios de leveza, portabilidade e facilidade de manutenção, considerando que o sistema deve operar em ambientes

industriais com possíveis restrições de conectividade e recursos computacionais limitados nos dispositivos dos operadores. A Tabela 3 resume as tecnologias utilizadas, suas respectivas funções no sistema e a justificativa técnica para cada escolha.

Tecnologia	**Função**	**Justificativa**
HTML5	Estrutura semântica	Padrão web moderno, suporte nativo a elementos multimídia
CSS3 + Grid/Flexbox	Layout responsivo	Adaptação automática a diferentes resoluções sem frameworks pesados
JavaScript (vanilla)	Lógica de interação	Sem dependências externas, execução rápida, fácil manutenção
Server-Sent Events	Telemetria em tempo real	Comunicação unidirecional eficiente, menor overhead que WebSocket
Fetch API	Requisições assíncronas	API nativa dos navegadores, suporte a Promises, código mais limpo
Python Flask	Backend web	Framework leve, integração direta com o SDK da Unitree

Tabela 3 – Stack tecnológico da interface web

A escolha por tecnologias nativas do navegador (sem frameworks como React, Vue ou Angular) teve como objetivo:

- Reduzir o tamanho da aplicação e o tempo de carregamento inicial;
- Eliminar etapas de build/transpilação durante o desenvolvimento;
- Facilitar a manutenção e debug por desenvolvedores com conhecimento básico de web;
- Garantir compatibilidade com navegadores embarcados ou dispositivos com recursos limitados.

Benefícios da arquitetura web

A implementação da interface como aplicação web (em vez de aplicativo desktop ou mobile nativo) trouxe vantagens práticas significativas:

1. **Acesso multiplataforma**: operação possível via desktop (Windows, macOS, Linux), tablets e smartphones, sem instalação de software;
2. **Atualizações centralizadas**: alterações no código são aplicadas apenas no servidor (Jetson), refletindo automaticamente para todos os clientes conectados;
3. **Baixa latência de rede**: em rede local (Wi-Fi 6 ou Ethernet), a comunicação entre navegador e Jetson apresenta latência inferior a 10 ms, adequada para teleoperação;
4. **Isolamento de responsabilidades**: frontend (apresentação) e backend (controle + IA) separados, permitindo desenvolvimento paralelo e testes independentes;
5. **Extensibilidade**: novos recursos (sensores adicionais, visualizações customizadas, dashboards de análise) podem ser incorporados sem recompilar ou redistribuir a aplicação.

Durante os testes de bancada, a interface permaneceu estável durante sessões contínuas de operação, com consumo de memória no navegador abaixo de 150 MB e CPU do cliente inferior a 5%, confirmando a eficiência da arquitetura adotada.

4.4 Resultados do treinamento do modelo YOLOv8-s

4.4.1 Curvas de treinamento e convergência

O modelo YOLOv8-s foi treinado utilizando o *Rust Detection Dataset* (Roboflow Universe, 2025) descrito no Capítulo 3, seguindo a divisão em treino/validação/teste gerada pelo Roboflow.

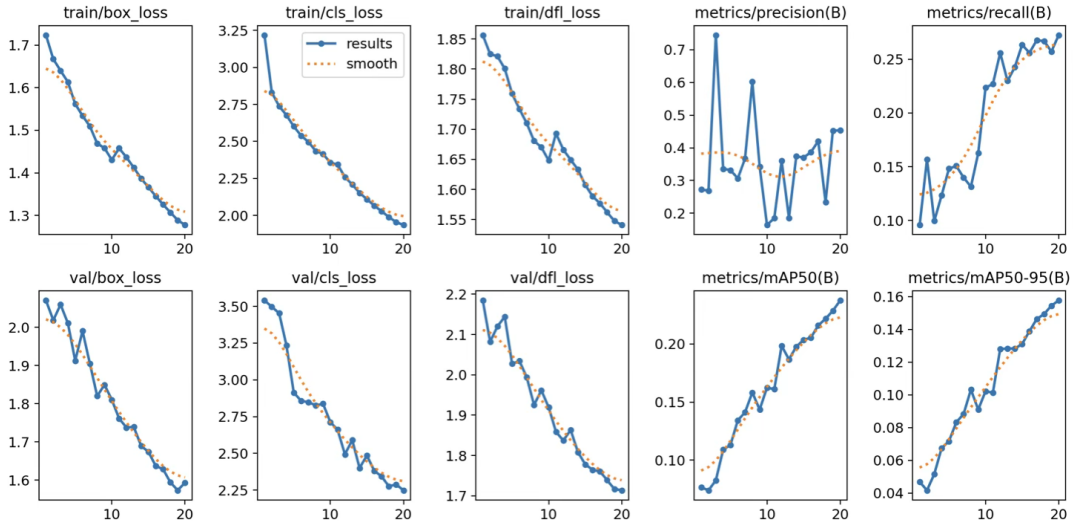

Figura 18 – Curvas de treinamento do modelo YOLOv8-s

Fonte: Ultralytics YOLOv8 (Ultralytics, 2025)

A Figura 18 apresenta o gráfico consolidado de treinamento gerado automaticamente pelo framework Ultralytics. Na parte superior da imagem, são exibidas as curvas de perda (loss) de treinamento e validação para cada componente: *box_loss* (erro de localização das caixas delimitadoras), *cls_loss* (erro de classificação das classes) e *dfl_loss* (distribution focal loss). Na parte inferior, são apresentadas as métricas de desempenho ao longo das épocas: precisão, recall, mAP@0.5 e mAP@0.5:0.95. As linhas azuis representam os valores de treinamento, enquanto as linhas laranja representam os valores de validação.

Observa-se que:

- As curvas de loss de treino e validação apresentam tendência decrescente e estável ao longo das épocas, indicando convergência do modelo.
- As métricas de desempenho (precision, recall, mAP@0.5 e mAP@0.5:0.95) aumentam significativamente nas primeiras épocas e tendem a se estabilizar a partir de um determinado ponto do treinamento, o que é compatível com um cenário de *fine-tuning* sobre pesos pré-treinados.

Esse comportamento sugere que o modelo conseguiu se adaptar ao domínio específico de corrosão, ainda que com níveis de desempenho moderados, dada a diversidade de classes e contextos presentes no dataset.

4.4.2 Métricas numéricas de desempenho por época

A Tabela 4 apresenta os valores numéricos das principais métricas de desempenho obtidas durante as 20 épocas de treinamento do modelo YOLOv8-s. Os dados consolidam tanto as perdas de treinamento e validação quanto as métricas de qualidade de detecção (precisão, recall e mAP).

Época	Precision	Recall	mAP@0.5	mAP@0.5:0.95	Train Loss	Val Loss	Tempo (s)
1	0,273	0,096	0,077	0,047	1,723	2,071	434
2	0,269	0,157	0,074	0,042	1,667	2,019	921
3	0,745	0,100	0,083	0,052	1,640	2,060	1.541
4	0,336	0,124	0,109	0,067	1,613	2,011	2.068
5	0,332	0,148	0,113	0,072	1,562	1,913	2.571
6	0,307	0,151	0,134	0,083	1,534	1,990	3.475
7	0,367	0,140	0,141	0,088	1,509	1,906	3.971
8	0,602	0,131	0,158	0,103	1,469	1,820	4.508
9	0,342	0,163	0,144	0,091	1,458	1,850	4.977
10	0,164	0,224	0,162	0,102	1,430	1,811	5.720
11	0,185	0,227	0,161	0,101	1,458	1,761	6.238
12	0,360	0,256	0,198	0,128	1,436	1,738	6.773
13	0,186	0,230	0,187	0,128	1,412	1,740	7.300
14	0,374	0,243	0,198	0,128	1,387	1,690	7.732
15	0,370	0,263	0,204	0,131	1,366	1,674	8.168
16	0,387	0,256	0,206	0,139	1,345	1,637	8.746
17	0,421	0,268	0,216	0,146	1,326	1,629	9.208
18	0,234	0,267	0,222	0,149	1,307	1,595	9.711
19	0,452	0,257	0,229	0,155	1,289	1,574	10.235
20	0,454	0,272	0,238	0,158	1,278	1,594	10.672

Tabela 4 – Resultados do treinamento do modelo YOLOv8-s por época

Observações sobre os resultados numéricos:

- **Convergência das perdas**: A perda de treinamento (Train Loss) decresceu de forma consistente de 1,723 (época 1) para 1,278 (época 20), indicando aprendizado efetivo. A perda de validação (Val Loss) também apresentou tendência de queda, passando de 2,071 para 1,594, embora com maior variabilidade.
- **Evolução da precisão**: A métrica de precisão apresentou oscilações ao longo do treinamento, variando entre 0,164 e 0,745. Ao final, estabilizou-se em torno de 0,45, o que representa cerca de 45% de detecções corretas entre todas as predições positivas.
- **Crescimento do recall**: O recall aumentou gradualmente de 0,096 (época 1) para 0,272 (época 20), indicando que o modelo progressivamente melhorou sua capacidade de encontrar as instâncias de corrosão existentes nas imagens.
- **mAP@0.5**: A métrica mAP@0.5 (mean Average Precision com limiar de IoU de 0,5) evoluiu de 0,077 para 0,238, demonstrando ganho significativo na qualidade geral das detecções.

- **mAP@0.5:0.95**: O mAP considerando múltiplos limiares de IoU (0,5 a 0,95) atingiu 0,158 na última época, valor moderado que reflete a dificuldade do modelo em produzir caixas delimitadoras com sobreposição muito precisa em relação às anotações de referência.
- **Tempo de treinamento**: O tempo acumulado por época aumentou progressivamente devido ao processamento completo de todo o dataset a cada iteração, totalizando aproximadamente 10.672 segundos (cerca de 3 horas) para as 20 épocas completas.

Os valores obtidos confirmam que o modelo atingiu um patamar de desempenho utilizável para o contexto de prova de conceito, embora ainda existam oportunidades de melhoria por meio de técnicas de otimização, aumento de dados e refinamento da base de treinamento com exemplos mais específicos do domínio de aplicação.

4.4.3 Distribuição de classes e características espaciais do dataset

Antes de analisar o desempenho do modelo em termos de matrizes de confusão e curvas de precisão-recall, é importante caracterizar a distribuição das classes e a organização espacial das anotações no conjunto de dados utilizado. A Figura 19 apresenta uma análise completa do *Rust Detection Dataset* (Roboflow Universe, 2025), consolidando em quatro painéis as características fundamentais do dataset.

Na Figura 19, o painel superior esquerdo exibe um gráfico de barras com a contagem de instâncias por classe, evidenciando o desbalanceamento característico de datasets de defeitos industriais. O painel superior direito apresenta um mapa de calor (heatmap) indicando a distribuição espacial acumulada dos centros das *bounding boxes* nas imagens, revelando onde as anotações tendem a se concentrar. Os dois painéis inferiores apresentam scatter plots que relacionam, respectivamente, as coordenadas x e y dos centros das caixas (esquerda) e as dimensões width e height das caixas (direita), permitindo visualizar padrões de posicionamento e tamanho das anotações.

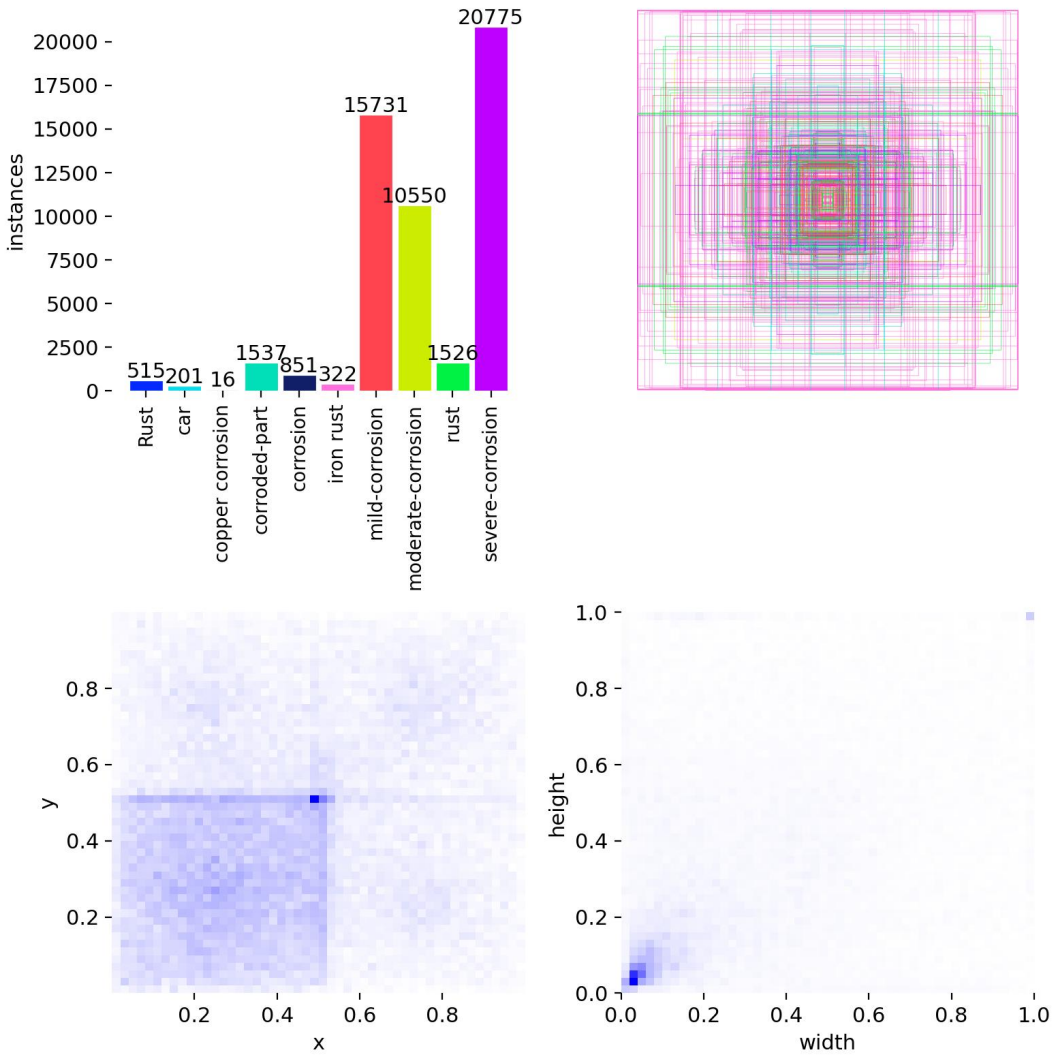

Figura 19 – Distribuição de classes e características espaciais das anotações no dataset

Fonte: Roboflow Universe (Roboflow Universe, 2025)

A análise do gráfico superior esquerdo revela a distribuição de instâncias por classe no dataset completo (treino + validação + teste). Observa-se um forte desbalanceamento entre as classes:

- **Classes dominantes**: *severe-corrosion* apresenta 20.775 instâncias, seguida por *mild-corrosion* (15.731) e *moderate-corrosion* (10.550). Essas três classes de severidade de corrosão representam a maior parte do conjunto de dados, o que é adequado ao objetivo do projeto.
- **Classes intermediárias**: *corroded-part* (1.537), *rust* (1.526), *corrosion* (651), *Rust* (515) e *iron rust* (322) aparecem com frequências moderadas, contribuindo para a diversidade de padrões visuais de deterioração.
- **Classes minoritárias**: *car* (201) e *copper corrosion* (16) são significativamente menos frequentes. A classe *car* representa um contexto de cena (presença de veículo), não

uma anomalia estrutural em si, enquanto *copper corrosion* é um tipo específico e raro de corrosão.

Esse desbalanceamento é comum em datasets de detecção de defeitos e reflete a realidade de que alguns tipos de corrosão (como a severa) são mais documentados por serem mais críticos. Durante o treinamento, o YOLOv8 utiliza estratégias internas para lidar com esse desbalanceamento, mas o modelo tende naturalmente a apresentar melhor desempenho nas classes mais representadas.

O mapa de calor superior direito ilustra a distribuição espacial acumulada de todas as *bounding boxes* nas imagens do dataset, revelando onde as anotações tendem a se concentrar. Observa-se uma concentração central, indicando que a maior parte das regiões de interesse foi capturada no centro das imagens, o que é esperado em fotografias de inspeção focadas.

Os dois gráficos inferiores apresentam a distribuição das coordenadas dos centros das *bounding boxes* (gráfico à esquerda, eixos x e y) e a distribuição das dimensões das caixas (gráfico à direita, eixos width e height). Essas visualizações mostram que:

- As caixas estão distribuídas por toda a área da imagem, embora com maior densidade em regiões centrais;
- A maioria das *bounding boxes* possui dimensões pequenas a moderadas, refletindo a presença de defeitos localizados e regiões específicas de corrosão, em vez de grandes áreas uniformemente deterioradas.

Essa caracterização espacial é relevante porque influencia o comportamento do modelo em campo: regiões de corrosão muito pequenas ou localizadas nas bordas da imagem podem ser mais difíceis de detectar, especialmente sob condições de enquadramento ou iluminação não ideais. Por outro lado, a boa cobertura espacial das anotações ajuda o modelo a generalizar para diferentes posições dentro do campo de visão da câmera.

4.4.4 Análise por classe e matrizes de confusão

Para uma análise mais detalhada do comportamento do modelo por classe, foram geradas as matrizes de confusão apresentadas nas Figuras 20 e 21. A Figura 20 apresenta a matriz de confusão normalizada, onde cada célula indica a proporção de predições para cada combinação de classe verdadeira (eixo vertical) e classe predita (eixo horizontal). Valores próximos a 1,0 na diagonal principal indicam alta taxa de acerto, enquanto valores elevados fora da diagonal indicam confusão entre classes. Observa-se que as classes de corrosão com maior representatividade no dataset (severe-corrosion, mild-corrosion, moderate-corrosion) apresentam valores mais elevados na diagonal, indicando melhor desempenho de classificação.

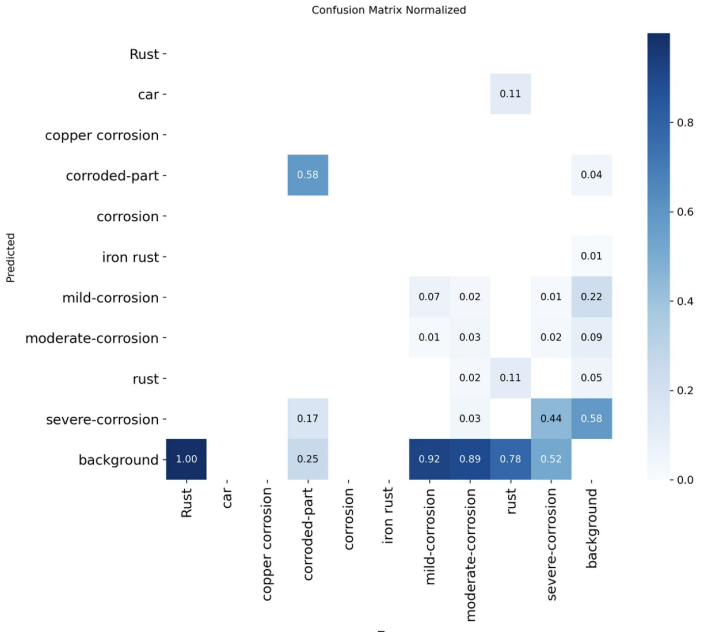

Figura 20 – Matriz de confusão normalizada

Fonte: Ultralytics YOLOv8 (Ultralytics, 2025)

A Figura 21 apresenta a matriz de confusão com valores absolutos, permitindo visualizar o número exato de predições em cada categoria. Esta representação complementa a matriz normalizada ao evidenciar o volume de amostras em cada classe, revelando o impacto do desbalanceamento do dataset nas métricas de desempenho.

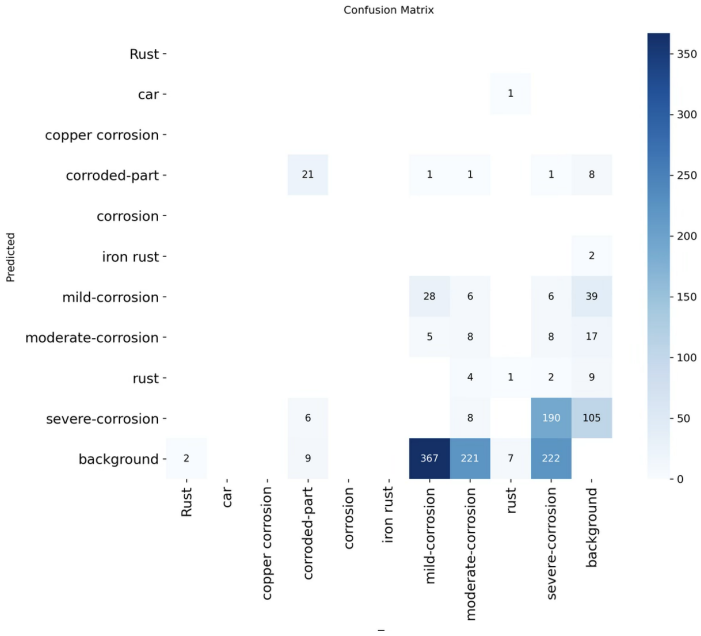

Figura 21 – Matriz de confusão (valores absolutos)

Fonte: Ultralytics YOLOv8 (Ultralytics, 2025)

Em linhas gerais, nota-se que:

- As classes associadas a corrosão mais evidente (por exemplo, regiões severamente corroídas ou partes inteiras comprometidas) tendem a concentrar a maior parte das detecções corretas.
- Há confusão entre classes que representam níveis diferentes de corrosão (mild, moderate, severe), o que é esperado, pois a transição entre esses estágios nem sempre é nítida visualmente, e muitas imagens apresentam regiões com gradação contínua de deterioração.
- A separação entre áreas realmente corroídas e regiões de fundo não corrosivo é, em geral, satisfatória, limitando a quantidade de falsos positivos em áreas saudáveis.

As curvas de precisão, recall, F1-score e precisão-recall por classe complementam a análise ao indicar como o desempenho varia em função do limiar de confiança adotado na inferência. Esses gráficos permitem escolher, para a aplicação embarcada, um compromisso entre precisão e recall adequado ao uso prático: por exemplo, aceitar mais falsos positivos em troca de menos falsos negativos em um cenário de inspeção preventiva.

A Figura 22 apresenta as curvas de Precisão-Recall para cada classe do modelo. Cada linha colorida representa uma classe específica, e a área sob a curva (AUC) indica o desempenho geral da detecção para aquela classe. Classes com curvas mais próximas do canto superior direito apresentam melhor equilíbrio entre precisão e recall. A linha tracejada indica a média de todas as classes (mAP@0.5).

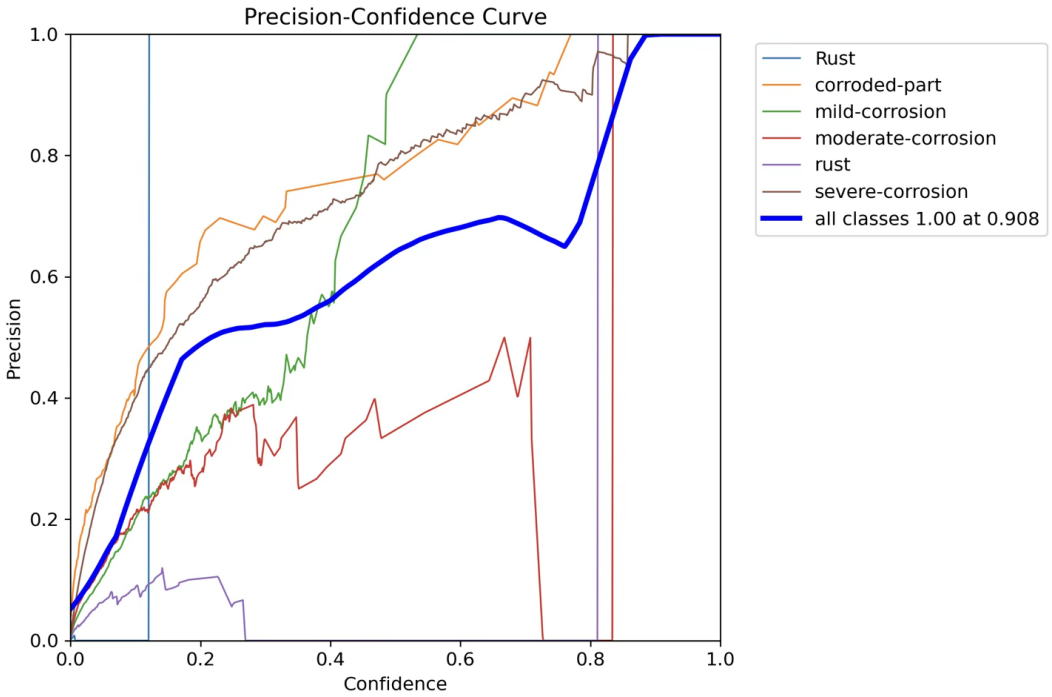

Figura 22 – Curvas de Precisão-Recall por classe (BoxPR)

A Figura 23 mostra como a precisão do modelo varia em função do limiar de confiança utilizado para filtrar as detecções. Limiares mais altos resultam em maior precisão (menos

falsos positivos), mas podem aumentar os falsos negativos. A análise desta curva auxiliou na definição do limiar padrão utilizado na interface CCO.

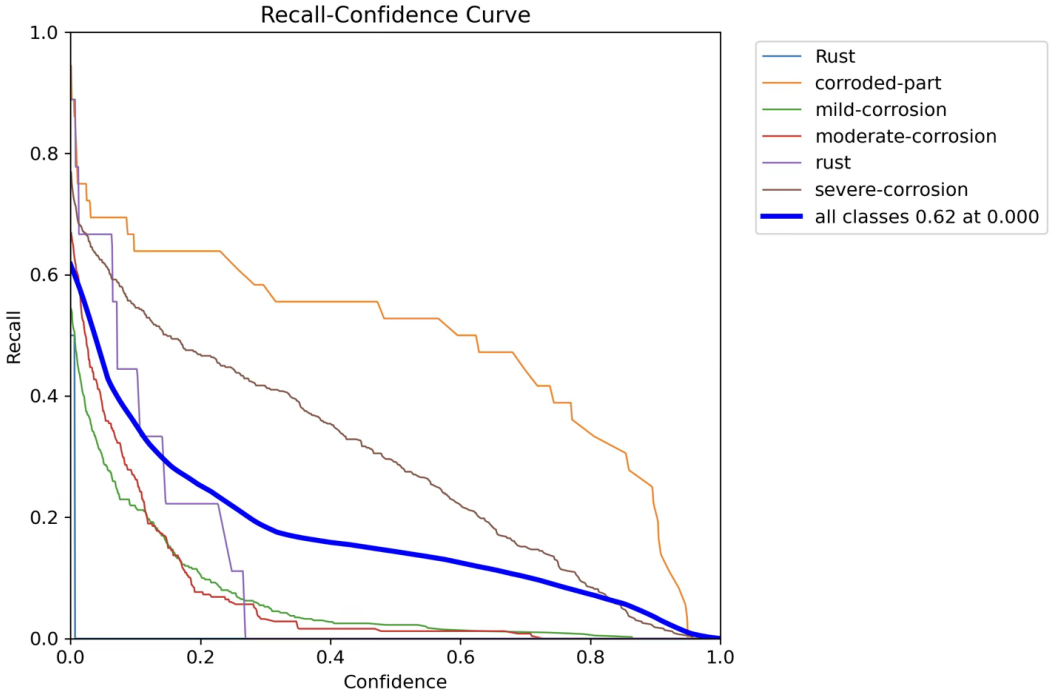

Figura 23 – Curva de Precisão (BoxP) por limiar de confiança

A Figura 24 apresenta a curva de Recall em função do limiar de confiança. Como esperado, o recall diminui à medida que o limiar aumenta, pois menos detecções são aceitas. Para aplicações de inspeção onde é crítico não perder defeitos (alta recall), pode ser necessário utilizar limiares mais baixos, aceitando maior taxa de falsos positivos.

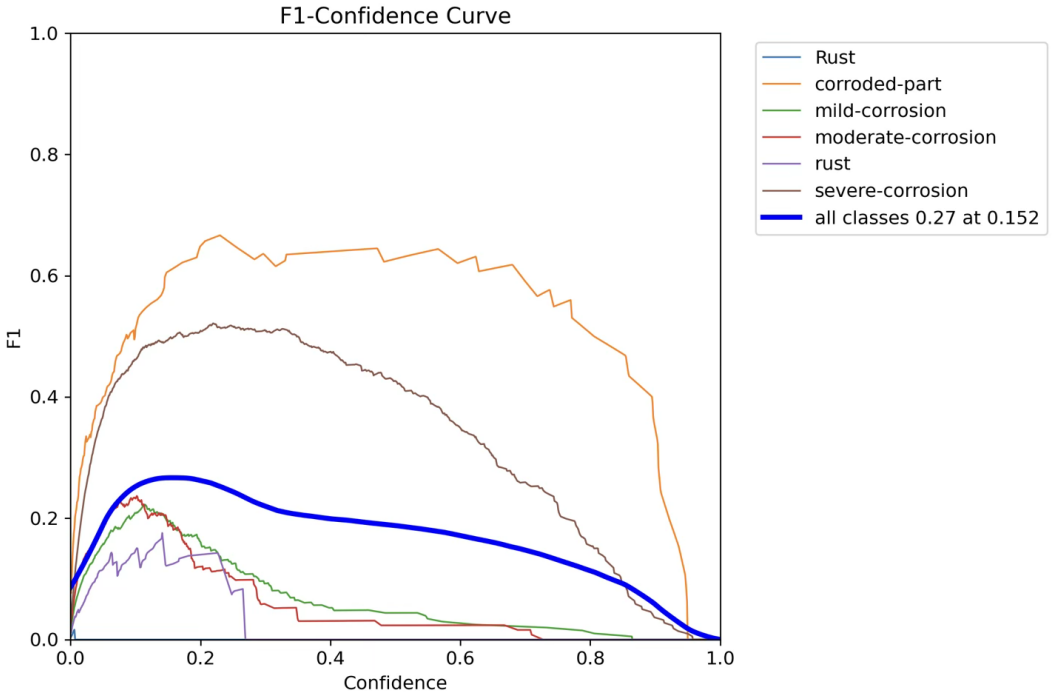

Figura 24 – Curva de Recall (BoxR) por limiar de confiança

A Figura 25 apresenta a curva de F1-Score, que representa a média harmônica entre precisão e recall. O ponto máximo desta curva indica o limiar de confiança que oferece o melhor equilíbrio entre evitar falsos positivos e não perder detecções verdadeiras. Este valor foi utilizado como referência para configuração padrão do sistema.

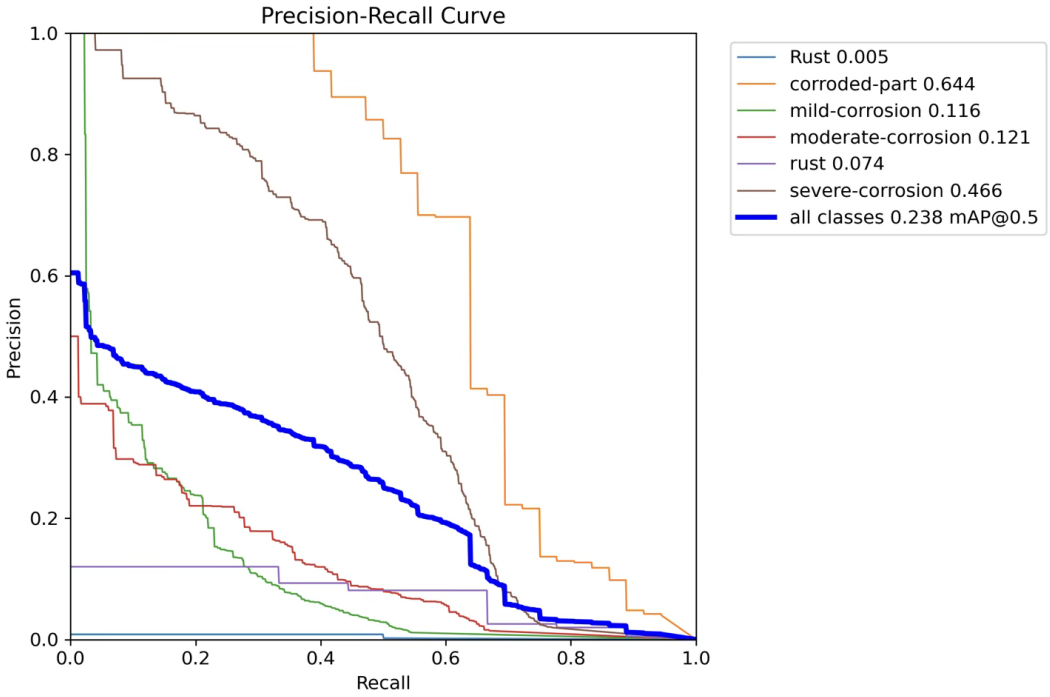

Figura 25 – Curva de F1-Score (BoxF1) por limiar de confiança

Fonte: Ultralytics YOLOv8 (Ultralytics, 2025)

4.4.5 Exemplos visuais de detecções no conjunto de validação

Para ilustrar o comportamento prático do modelo YOLOv8-s treinado, são apresentadas a seguir visualizações das detecções realizadas em imagens do conjunto de validação. Essas figuras mostram como o modelo identifica e classifica diferentes tipos e severidades de corrosão em diversos contextos visuais.

As Figuras 26 e 27 apresentam grids de imagens de validação com as *bounding boxes* detectadas pelo modelo, identificadas apenas por números de classe. Essas visualizações permitem observar a cobertura espacial das detecções e a capacidade do modelo de localizar múltiplas instâncias em uma mesma imagem.

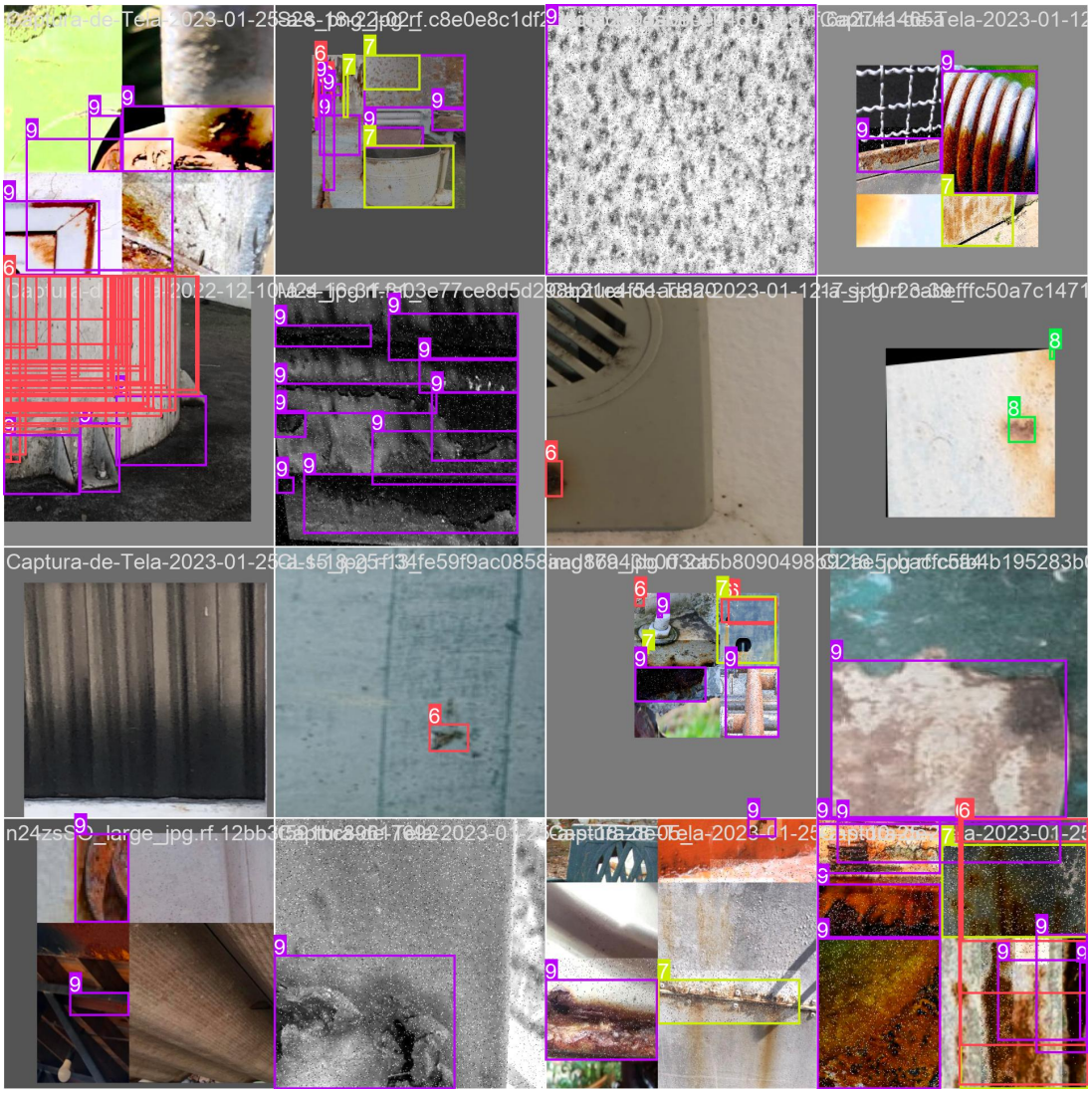

Figura 26 – Batch de validação 1: detecções com identificação numérica de classes

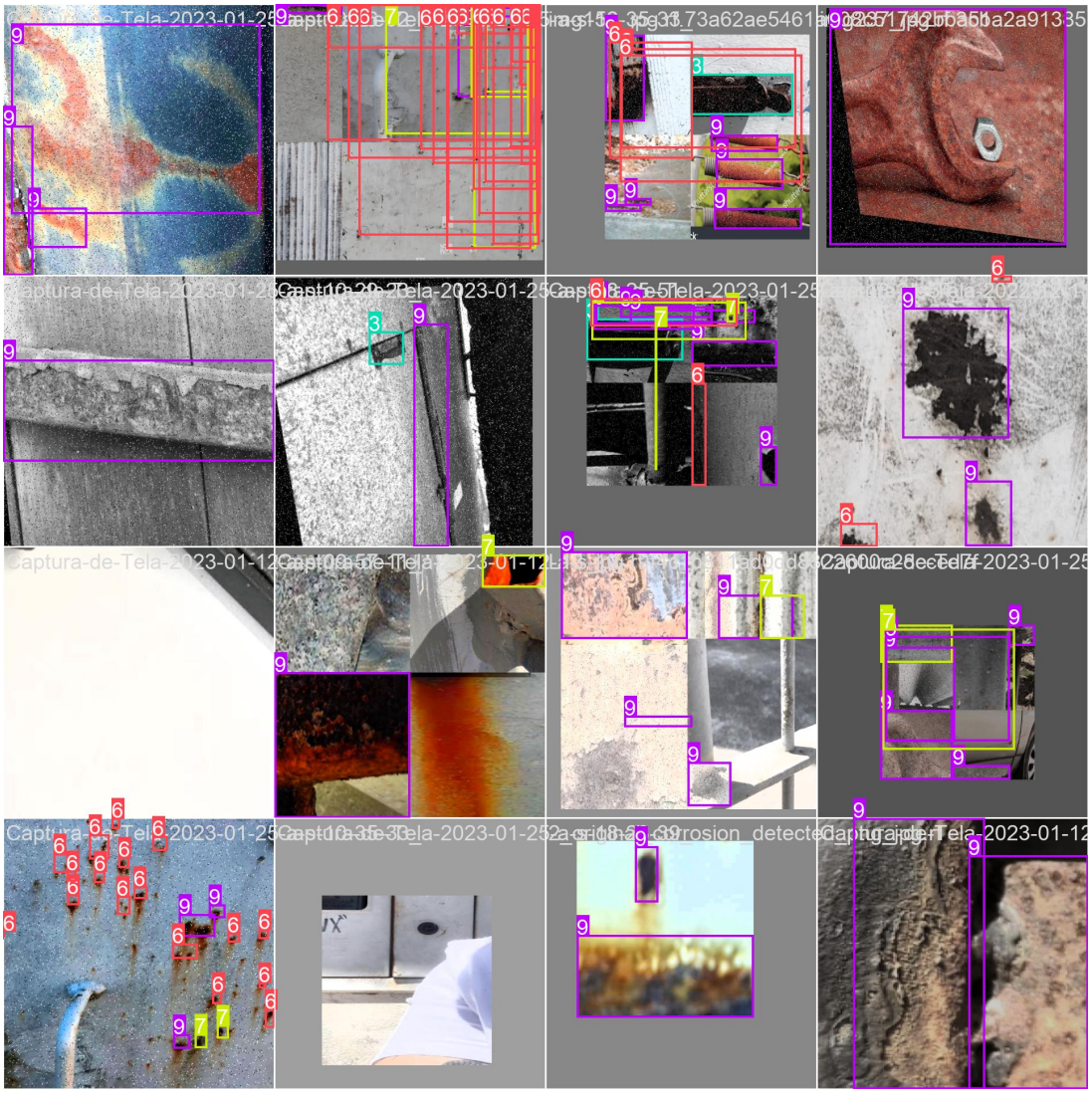

Figura 27 – Batch de validação 2: detecções com identificação numérica de classes

As Figuras 28 e 29 exibem as mesmas imagens de validação, mas com os rótulos textuais das classes sobrepostos às *bounding boxes*. Observa-se que o modelo consegue distinguir entre diferentes níveis de severidade (*mild-corrosion*, *moderate-corrosion*, *severe-corrosion*), além de identificar contextos específicos como *corroded-part* e *iron rust*.

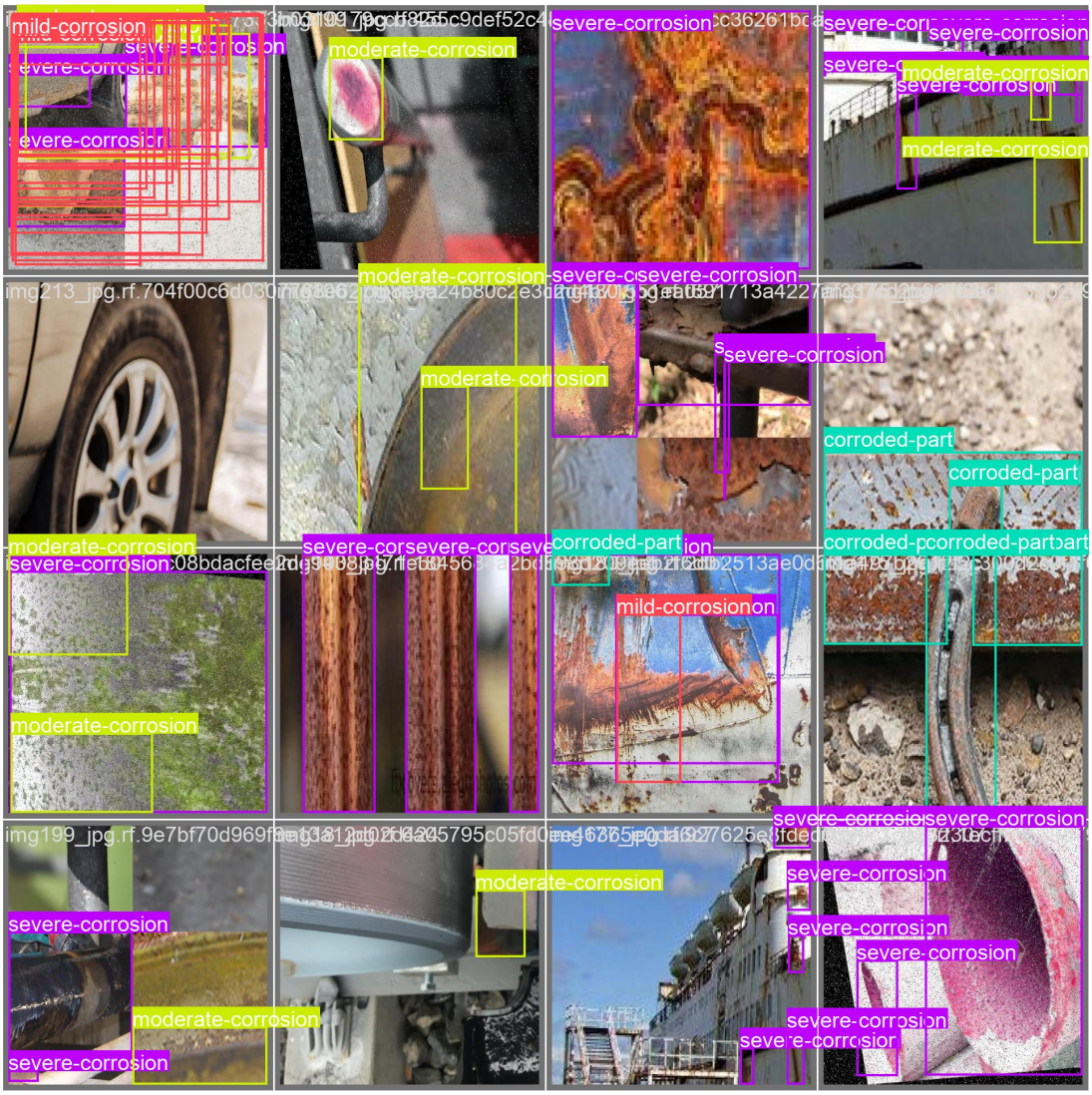

Figura 28 – Batch de validação 3: detecções com rótulos de classe

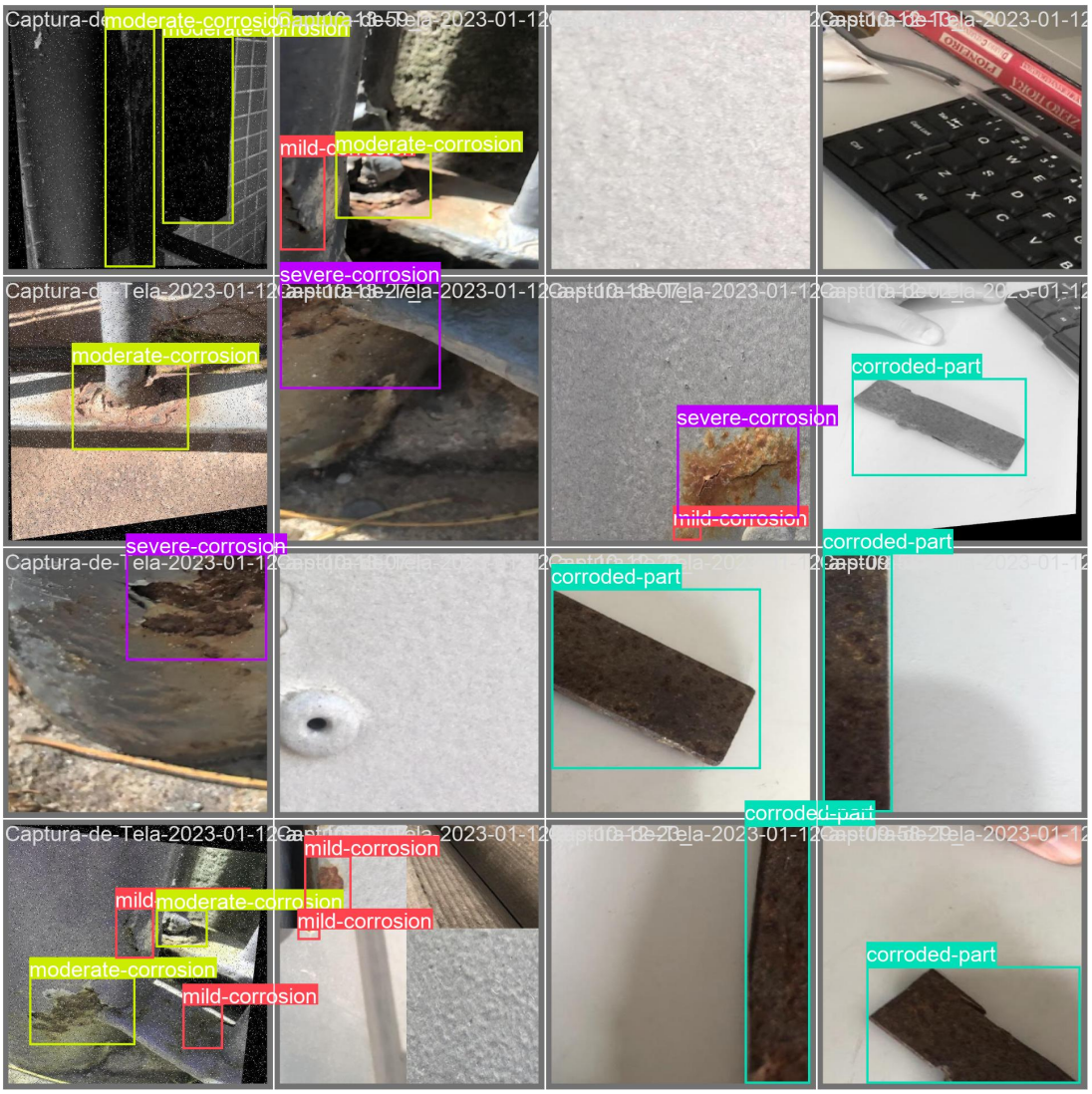

Figura 29 – Batch de validação 4: detecções com rótulos de classe

Figura 30 – Batch de validação 5: detecções com rótulos de classe

As Figuras 31, 32 e 33 complementam a análise ao incluir, além dos rótulos de classe, os valores de confiança (confidence scores) associados a cada detecção. Esses valores, expressos como probabilidades entre 0 e 1, indicam o grau de certeza do modelo em relação à presença e à classificação de cada instância detectada.

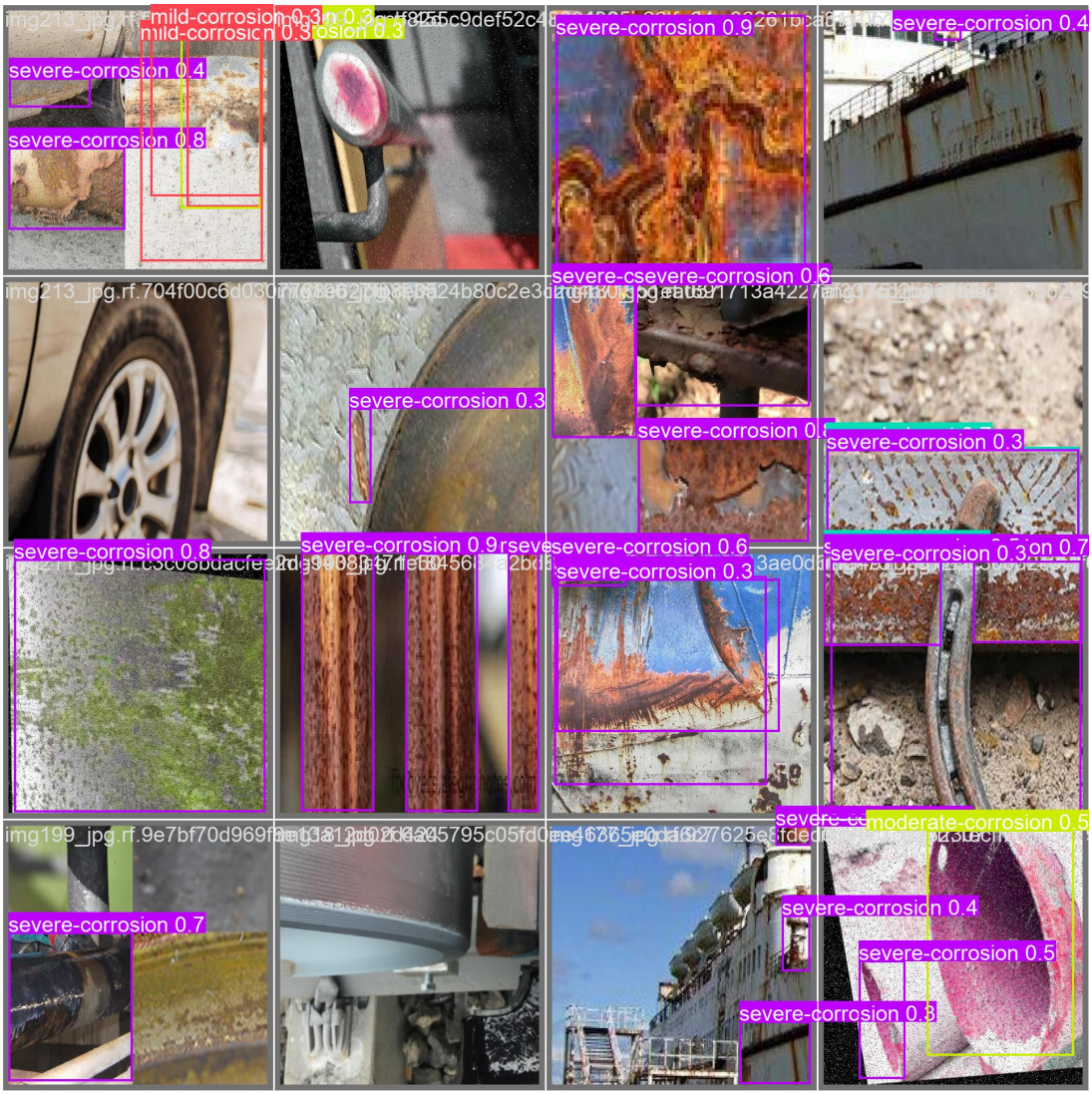

Figura 31 – Batch de validação com rótulos e scores de confiança (1)

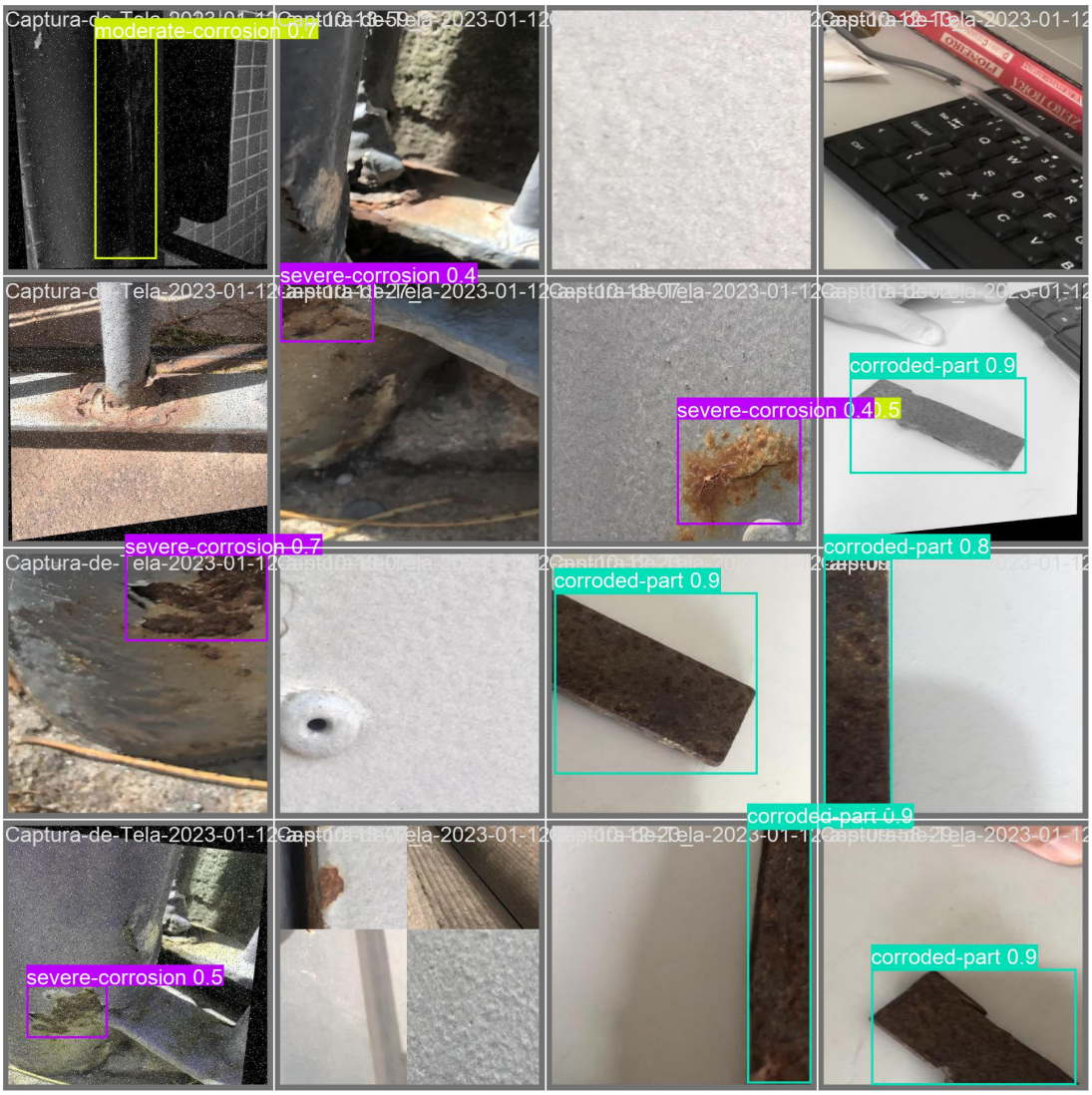

Figura 32 – Batch de validação com rótulos e scores de confiança (2)

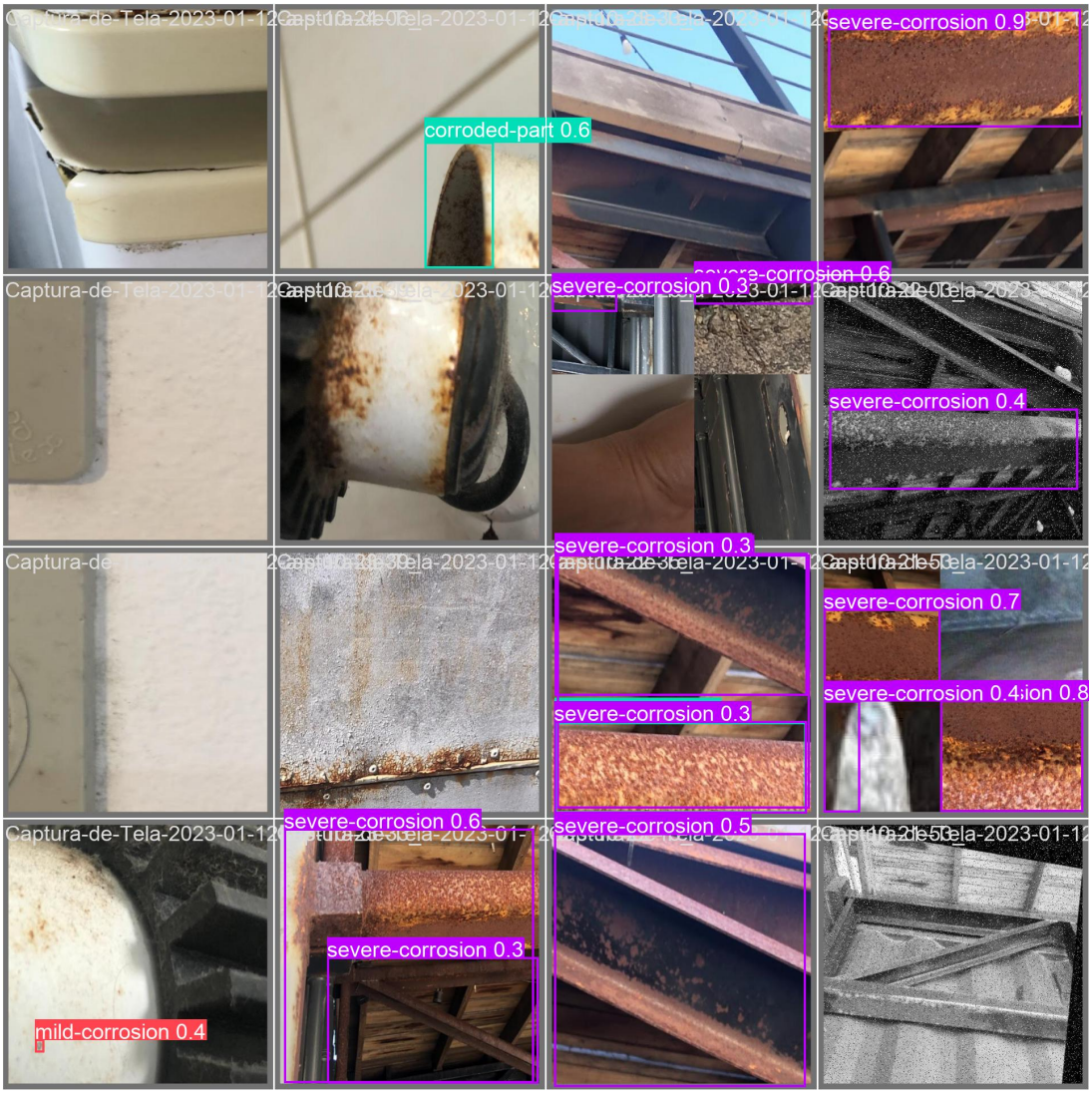

Figura 33 – Batch de validação com rótulos e scores de confiança (3)

Análise qualitativa das detecções visualizadas:

- **Diversidade de contextos**: As imagens de validação abrangem diferentes tipos de superfícies metálicas (tubos, chapas, estruturas industriais, peças automotivas), ambientes (internos, externos, áreas molhadas) e condições de iluminação, demonstrando a capacidade de generalização do modelo.
- **Detecção de múltiplas instâncias**: Em várias imagens, o modelo identifica corretamente múltiplas regiões de corrosão na mesma cena, inclusive com diferentes níveis de severidade coexistindo (por exemplo, áreas de corrosão leve próximas a áreas severamente deterioradas).
- **Classificação de severidade**: O modelo demonstra habilidade em distinguir entre corrosão leve (mild), moderada (moderate) e severa (severe), o que é relevante para priorização de intervenções de manutenção.
- **Confiança das predições**: Os scores de confiança apresentados variam predominante-

mente entre 0,3 e 0,9. Detecções com confiança acima de 0,7 geralmente correspondem a regiões bem definidas de corrosão, enquanto valores entre 0,3 e 0,5 frequentemente aparecem em áreas de transição, manchas ambíguas ou bordas de regiões corroídas.

- **Limitações observadas**: Nota-se que em algumas imagens ocorrem sobreposições de caixas (especialmente em áreas de corrosão extensa), indicando que o modelo pode gerar múltiplas detecções para uma mesma região. Além disso, em superfícies muito texturizadas ou com padrões visuais complexos, observa-se ocasionalmente a presença de falsos positivos ou confusão entre classes de severidade próximas.

Essas visualizações confirmam que o modelo treinado é capaz de realizar detecções estruturadas e semanticamente coerentes em contextos variados, embora ainda apresente oportunidades de melhoria em termos de precisão de localização, supressão de detecções redundantes e robustez em cenários visualmente desafiadores.

4.5 Ensaios em ambiente de bancada

4.5.1 Cenário de teste e protocolo experimental

A validação prática do sistema completo foi realizada por meio de ensaios de bancada em ambiente interno controlado, configurado especificamente para avaliar a integração entre o robô Go2, a câmera Intel RealSense D435i, o modelo de IA YOLOv8-s e a interface web CCO. Diferentemente de testes de navegação autônoma ou locomoção (já validados em trabalhos anteriores sobre a plataforma Go2), o foco desta etapa experimental foi a avaliação do pipeline de visão computacional embarcada para detecção de corrosão.

A Figura 34 apresenta o cenário de teste configurado para os ensaios de bancada. Na imagem, observa-se o robô Go2 posicionado diante de chapas metálicas com diferentes graus de corrosão, representando superfícies típicas encontradas em estruturas industriais e de saneamento. O ambiente de laboratório foi organizado para permitir o controle de variáveis como distância entre câmera e superfície, ângulo de observação e condições de iluminação. Ao fundo, visualiza-se o computador do operador executando a interface web CCO, que recebe o fluxo de vídeo com as detecções do modelo de IA em tempo quase real.

Figura 34 – Cenário de teste com chapas metálicas corroídas

O robô Unitree Go2 foi posicionado em frente a:

- uma chapa de aço com regiões de corrosão visível;
- outros itens metálicos com ferrugem e manchas semelhantes à corrosão.

O robô foi teleoperado apenas para ajuste de posição e enquadramento, permanecendo parado na maioria dos testes, com a câmera RealSense apontada para as superfícies de interesse a curta distância. A Jetson Orin NX processava os frames em tempo quase real, executando o modelo YOLOv8-s e enviando as detecções (caixas e rótulos) para a interface web.

O objetivo desses ensaios foi verificar qualitativamente:

- se o modelo conseguia destacar, na imagem, as regiões com padrão de ferrugem/corrosão;
- se a sobreposição das detecções na interface era estável e acompanhava pequenas mudanças de enquadramento;
- e se a latência percebida pelo operador era compatível com o uso em inspeção visual assistida.

Para registro, foram coletados principalmente:

- capturas de tela da interface web com as detecções sobrepostas;
- imagens estáticas de alguns quadros representativos;
- anotações observacionais sobre casos de acerto, falsos positivos e situações em que o modelo não detectou regiões corroídas.

Esses ensaios fornecem uma primeira evidência de funcionamento do pipeline embarcado em condições simples de teste, servindo como base para validações futuras em cenários mais complexos de inspeção.

A Figura 35 apresenta um exemplo representativo das detecções realizadas pelo modelo YOLOv8-s durante os ensaios de bancada. Na imagem, observam-se as *bounding boxes* sobrepostas às regiões identificadas como corrosão, com os respectivos rótulos de classe (indicando o tipo ou severidade da corrosão detectada) e os valores de confiança (confidence scores) associados a cada detecção. As regiões destacadas em cores distintas representam diferentes

classificações do modelo: corrosão severa, corrosão moderada e corrosão leve. Este tipo de visualização é exibido em tempo quase real na interface CCO durante a operação do sistema, permitindo ao operador identificar rapidamente as áreas que requerem atenção durante a inspeção.

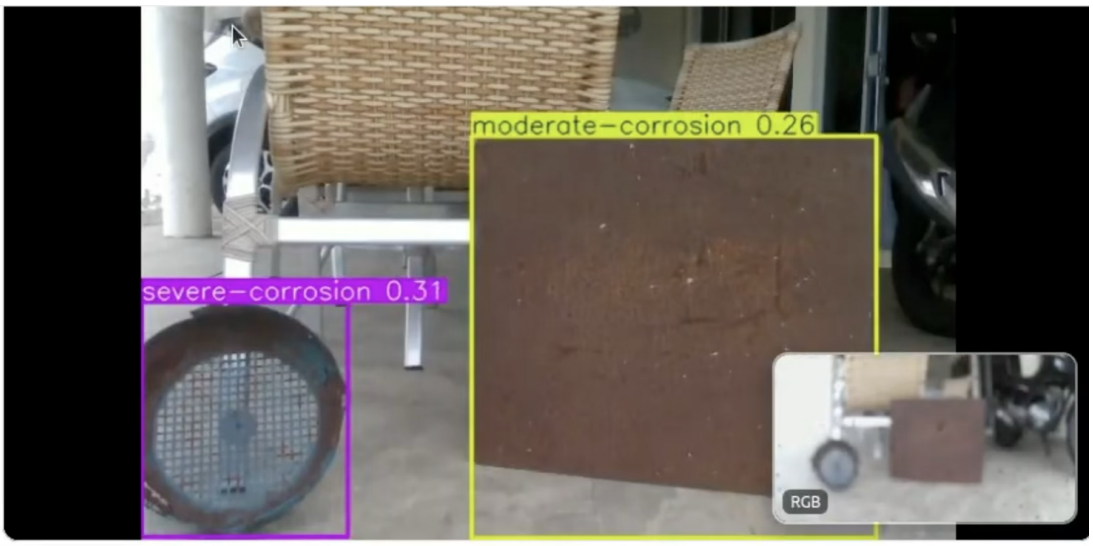

Figura 35 – Resultado da detecção de corrosão em ensaio de bancada

4.5.2 Desempenho da detecção em ensaios de bancada

Nos ensaios de bancada descritos na Seção 4.5.1, o modelo YOLOv8-s embarcado na Jetson Orin NX mostrou-se capaz de:

- destacar visualmente as regiões com padrão de ferrugem/corrosão nas chapas metálicas e objetos utilizados nos testes;
- manter detecções consistentes quando o enquadramento da câmera era ajustado levemente (pequenos deslocamentos ou aproximações do robô);
- identificar, em vários casos, áreas de corrosão difusas ou irregulares, auxiliando o operador a localizar rapidamente as regiões de interesse.

Também foram observadas limitações importantes:

- ocorrência de falsos positivos em manchas não corrosivas (sujeira, tinta descascada ou desgaste superficial);
- perda de detecção em áreas de corrosão muito pequenas ou parcialmente fora do campo de visão.

Uma limitação relevante diz respeito ao desempenho em tempo real do pipeline de IA. Durante os testes, o fluxo de vídeo processado pelo YOLOv8-s na Jetson apresentou baixa taxa de quadros e atraso perceptível (alguns segundos) entre o movimento real e a atualização das detecções na interface web. Esse gargalo de processamento fazia com que a imagem com as detecções “travasse” e não acompanhasse suavemente o movimento.

Por outro lado, o fluxo de vídeo da câmera RGB sem processamento de IA permaneceu fluido, sem atraso perceptível. Dessa forma, a operação do robô em si não foi comprometida: o operador podia conduzir o Go2 normalmente usando a imagem "limpa", enquanto a saída do modelo de IA funcionava mais como um retorno assíncrono de análise.

Em síntese, os testes indicam que o modelo é útil como ferramenta de pré-filtragem, chamando atenção do operador para regiões suspeitas, mas ainda apresenta limitações tanto em acurácia (falsos positivos/negativos) quanto em desempenho em tempo real. Essa questão de desempenho é retomada na Seção 4.6 e 4.7, como oportunidade de melhoria futura do sistema.

4.6 Avaliação de latência e desempenho embarcado

A avaliação de desempenho embarcado concentrou-se no comportamento do pipeline de IA rodando na Jetson Orin NX e no impacto desse processamento sobre a visualização do operador.

Nos ensaios de bancada, foram observados dois fluxos distintos de vídeo:

- o fluxo processado pelo modelo YOLOv8-s (RealSense $\rightarrow$ Jetson $\rightarrow$ IA $\rightarrow$ interface web);
- o fluxo RGB "limpo", exibido na interface sem passar pelo modelo de IA.

Na prática verificou-se que:

- o fluxo com IA apresentou taxa de quadros reduzida e atraso perceptível, com atrasos de alguns segundos entre o movimento real e a atualização da imagem com as detecções na interface. Em determinados momentos, a imagem parecia "congelar" e atualizava em blocos, caracterizando um gargalo de processamento na Jetson Orin NX para esse modelo e configuração;
- o fluxo RGB sem IA manteve-se fluido, sem atraso visual relevante, permitindo ao operador controlar o robô normalmente e posicionar a câmera com boa resposta.

Assim, do ponto de vista de teleoperação, o sistema mostrou-se utilizável, pois o operador se guia pela câmera sem IA. Contudo, do ponto de vista de visualização das detecções em tempo quase real, o desempenho atual é limitado: o pipeline completo (captura $\rightarrow$ IA $\rightarrow$ exibição) ainda não atinge a latência desejável para uso contínuo como camada principal de navegação.

Os ensaios foram conduzidos com sessões de curta duração, suficientes para verificar o funcionamento do pipeline, mas não para caracterizar de forma precisa a autonomia energética sob carga de IA. Nessa fase, portanto, a análise de consumo de bateria e degradação de desempenho ao longo de longos períodos de inspeção é qualitativa: não foram observados desligamentos ou quedas bruscas de desempenho durante os testes realizados, mas uma caracterização detalhada de autonomia fica como trabalho futuro.

4.7 Limitações observadas e oportunidades de melhoria

A partir dos testes realizados, destacam-se as seguintes limitações e oportunidades de evolução do sistema:

• Desempenho em tempo real do modelo de IA

O principal gargalo identificado foi o tempo de processamento do YOLOv8-s na Jetson Orin NX, que resultou em baixa taxa de quadros e atraso de alguns segundos no fluxo com detecções. Como consequência, o overlay do modelo é mais adequado hoje como retorno auxiliar do que como visão principal do operador.

Entre as possibilidades de melhoria estão:

- uso de variantes mais leves do modelo (por exemplo, YOLOv8-n);
- técnicas de otimização como quantização, pruning e uso de TensorRT;
- redução da resolução de entrada ou da taxa de quadros enviada para o modelo.

• Cobertura limitada do domínio de inspeção

O modelo foi treinado em um dataset específico de corrosão (*Rust Detection Dataset* (Roboflow Universe, 2025)), com imagens coletadas em contextos gerais de indústria e automotivo. Nos testes de bancada, o comportamento foi razoável em chapas e peças metálicas, mas ainda não há validação sistemática em estruturas reais de campo (tubulações, galerias, estruturas complexas). A inclusão de imagens capturadas pelo próprio robô em inspeções reais e o fine-tuning contínuo do modelo tendem a melhorar a aderência ao domínio alvo.

• Sensibilidade a detalhes finos e falsos positivos

Em alguns casos, o modelo deixou de detectar áreas pequenas de corrosão ou confundiu manchas não corrosivas (sujeira, tinta descascada) com defeitos, gerando falsos positivos. Estratégias como ajuste de limiar de confiança por classe, refinamento de anotações e aumento de exemplos positivos/negativos nesses cenários específicos podem reduzir esse problema.

• Integração com funcionalidades avançadas de navegação

Na versão atual, a detecção atua de forma passiva: o operador escolhe manualmente onde apontar a câmera. Em versões futuras, as saídas da IA podem ser integradas a algoritmos de navegação e planejamento de trajetória, permitindo que o robô ajuste a rota para revisitar regiões suspeitas ou aproximar-se automaticamente de áreas com probabilidade alta de anomalia.

Esses pontos não invalidam a prova de conceito, mas indicam direções claras de trabalho futuro para transformar o protótipo em uma solução mais robusta e aderente a cenários industriais reais.

4.8 Síntese dos resultados experimentais

Os resultados apresentados ao longo deste capítulo permitem sintetizar o estado atual do sistema da seguinte forma:

- O protótipo físico (Unitree Go2 + Jetson Orin NX + Intel RealSense D435i + LiDAR L1) foi montado com sucesso, mantendo estabilidade mecânica e térmica adequada para ensaios de bancada e potencial para missões de inspeção de curta/média duração.
- O pipeline de software embarcado aquisição de dados, controle via SDK, execução do modelo YOLOv8-s e interface web opera de forma integrada. A teleoperação por meio da câmera RGB "limpa" mostrou-se fluida, enquanto o fluxo com IA apresentou atraso perceptível, caracterizando um gargalo de desempenho que precisa ser tratado em versões futuras.
- O modelo de IA treinado sobre o *Rust Detection Dataset* (Roboflow Universe, 2025) demonstrou ser capaz de destacar regiões com padrão de corrosão nas chapas e objetos metálicos utilizados nos testes, funcionando como ferramenta de apoio visual ao operador. Ao mesmo tempo, apresentou falsos positivos e perda de detecção em áreas pequenas, indicando necessidade de refinamento do treinamento e ampliação da base de imagens.
- Em ensaios de bancada, com o robô posicionado diante de peças com corrosão, o sistema foi capaz de executar detecção embarcada e exibir as marcações na interface, sem depender de conexão com a nuvem. A operação do robô não foi comprometida pelo gargalo da IA, graças ao uso simultâneo da câmera RGB sem processamento como referência principal para navegação.

De maneira geral, o trabalho demonstra que a combinação de robótica quadrúpede com processamento de visão embarcado é tecnicamente viável e já permite construir um protótipo funcional de inspeção assistida por IA. As limitações identificadas (sobretudo em desempenho em tempo real e especialização do modelo para o domínio específico de inspeção) configuram um ponto de partida claro para trabalhos futuros, incluindo otimização do pipeline de IA, coleta de dados em campo e integração com estratégias mais avançadas de navegação e planejamento de missões.

4.9 Exemplos de resultados gerados

Durante os ensaios de bancada, o sistema gerou três tipos principais de resultados: (i) imagens com as detecções da IA sobrepostas, (ii) registros de telemetria e (iii) sínteses textuais elaboradas a partir desses dados.

No primeiro caso, a interface web permitiu salvar capturas de tela nas quais as regiões identificadas pelo modelo YOLOv8-s aparecem destacadas por bounding boxes coloridas, acompanhadas do rótulo de classe (por exemplo, *mild-corrosion*, *severe-corrosion* ou *corroded-part*) e do valor de confiança associado. Essas imagens funcionam como "evidências visuais" da

inspeção, podendo ser anexadas a relatórios técnicos ou comparadas com registros de missões futuras.

Além das capturas de tela, o sistema registrou quadros individuais da câmera RealSense sem sobreposição de IA, juntamente com metadados básicos (data e hora da captura, identificação da missão, distância aproximada ao alvo). Esses arquivos permitem reanalisar posteriormente a cena original, com ou sem a aplicação de outros modelos ou ajustes de limiar de confiança.

Por fim, a partir dessas evidências visuais e dos registros de telemetria do robô (bateria, temperatura média, velocidade, modo de operação), foram produzidas sínteses textuais simples por missão de teste. Cada síntese descreve: o cenário ensaiado, os principais pontos de corrosão destacados pelo modelo, a ocorrência de falsos positivos/negativos observados pelo operador e eventuais dificuldades de visualização (iluminação, reflexos, enquadramento). Embora ainda não exista um módulo automático de geração de relatórios, esse procedimento demonstra como as saídas do sistema podem ser organizadas em documentos padronizados de inspeção em versões futuras da solução.

4.10 Problemas encontrados e soluções

• Baixa taxa de FPS e travamento no fluxo com IA

O principal problema observado nos testes foi o atraso e a baixa fluidez do vídeo processado pelo modelo YOLOv8-s na Jetson Orin NX. Em vários momentos, a imagem com as detecções apresentava atraso de alguns segundos em relação à cena real e atualizava em "saltos", dando a sensação de travamento.

Solução adotada: a teleoperação passou a ser feita prioritariamente usando o fluxo RGB "limpo", sem IA, que se manteve fluido e sem delay perceptível, garantindo controle seguro do robô. O overlay do YOLOv8-s foi tratado como retorno auxiliar, usado quando o robô está parado ou com pouco movimento. Como melhoria futura, foram identificadas ações como: testar versões mais leves do modelo (por exemplo YOLOv8-n), reduzir resolução/taxa de quadros de entrada e aplicar otimizações específicas (TensorRT, quantização).

• Limitações da conexão Wi-Fi e necessidade de rede mais estável

Em alguns momentos, a conexão via Wi-Fi entre a Jetson e o computador do operador apresentou instabilidades (oscilações de sinal e pequenos atrasos na interface web), especialmente em ambientes internos com outras redes ativas.

Solução adotada: sempre que possível, foi utilizada uma conexão de rede cabeada (Ethernet) por meio de adaptador conectado à Jetson, garantindo maior estabilidade e menor latência no acesso ao painel web. Além disso, foi mantida a possibilidade de acesso direto à Jetson por HDMI ou SSH para diagnóstico e recuperação, caso a rede sem fio apresente falhas.

• Falsos positivos em manchas não corrosivas

O modelo de IA registrou detecções indevidas em manchas de sujeira, tinta descascada ou desgaste superficial que não correspondiam a corrosão real.

Solução adotada: foi ajustado um limiar de confiança mais conservador para algumas classes, reduzindo parte dos falsos positivos, e adotada a prática de interpretação assistida: as regiões marcadas pela IA são tratadas como "suspeitas", cabendo ao operador confirmar ou descartar cada ocorrência. O refinamento do dataset e um novo ciclo de fine-tuning com exemplos reais do ambiente de aplicação foram identificados como próximos passos para reduzir esse tipo de erro.

4.11 Conformidade com a documentação do fabricante

• SDK Concepts: telemetria, controle e temporização

A implementação de controle e leitura de estados do Go2 baseou-se no documento SDK Concepts da Unitree. Foram seguidas as recomendações de inicialização, tratamento de mensagens e uso dos canais de comunicação, garantindo que os comandos enviados pela Jetson respeitassem a semântica e os limites previstos pelo fabricante.

• Payload: integração mecânica e elétrica do módulo Jetson e sensores

A instalação da Expansion Dock, da câmera Intel RealSense D435i e das conexões de alimentação/dados observou os limites de tensão, corrente e pinagem descritos no guia de Payload. Também foram respeitadas as orientações de posição do payload e roteamento de cabos, de modo a não comprometer o centro de massa do robô nem a integridade dos conectores.

• Module Update: atualização segura de firmware e serviços

Toda atualização de firmware do Go2 e dos serviços associados foi realizada com base no módulo Module Update, utilizando o fluxo oficial de download, verificação de integridade (checksum), instalação e reboot seguro. Após cada ciclo de atualização, os logs armazenados em /var/log/unitree foram inspecionados, e testes rápidos de comunicação e controle foram executados antes da retomada dos ensaios.

5 Discussão

Este capítulo apresenta uma análise crítica dos resultados obtidos, contextualizando-os em relação aos objetivos propostos, à hipótese de pesquisa e ao estado da arte na área de inspeção robótica assistida por inteligência artificial. São discutidas as implicações práticas dos resultados, comparando o sistema desenvolvido com métodos tradicionais de inspeção e outras tecnologias automatizadas. Além disso, são identificadas as limitações do trabalho e propostas direções para melhorias futuras, tanto em termos de hardware e software quanto de operação e usabilidade. Por fim, é analisada a relevância acadêmica e industrial da pesquisa, destacando as contribuições específicas e o potencial de aplicação da solução desenvolvida.

5.1 Interpretação geral dos resultados

Os resultados obtidos indicam que o sistema desenvolvido atingiu o objetivo principal de integrar um robô quadrúpede com visão de profundidade e IA embarcada para apoiar inspeções em ambientes confinados.

Do ponto de vista de integração, o conjunto formado por Unitree Go2 + Jetson Orin NX + Intel RealSense D435i + LiDAR L1 mostrou-se funcional: a montagem mecânica manteve a estabilidade do robô, a integração elétrica respeitou os limites de alimentação do fabricante e o pipeline de software (aquisição, SDK, IA e interface web) operou de forma coesa.

A interface CCO (Go2 EDU) permitiu ao operador teleoperar o robô, visualizar métricas de telemetria e acompanhar, ainda que com limitações, as detecções do modelo YOLOv8-s.

No nível da IA, o modelo treinado a partir do *Rust Detection Dataset* (Roboflow Universe, 2025) mostrou-se capaz de destacar áreas com padrão visual compatível com corrosão nas chapas de teste, funcionando como uma camada de apoio à inspeção visual.

Entretanto, a análise em bancada evidenciou a presença de falsos positivos (manchas não corrosivas identificadas como corrosão) e perda de detecção em áreas pequenas ou pouco nítidas, o que indica que o sistema, no estado atual, não substitui o julgamento humano, mas ajuda a direcionar a atenção do operador para regiões suspeitas.

Um ponto crítico observado foi o desempenho em tempo real do pipeline de IA: o fluxo de vídeo processado pelo YOLOv8-s apresentou atraso e baixa taxa de quadros, enquanto o fluxo RGB sem IA permaneceu fluido.

Na prática, isso fez com que o operador utilizasse a imagem “limpa” para conduzir o robô e tratasse a saída da IA como retorno auxiliar, especialmente com o robô parado ou com pouco movimento.

Assim, a interpretação geral é que o trabalho comprova a viabilidade técnica da abordagem e entrega um protótipo funcional, mas também deixa claro que ainda são necessárias otimizações

de modelo, refinamento de base de dados e melhorias de desempenho para que o sistema atinja o nível de maturidade exigido em operações industriais rotineiras.

5.2 Comparação com métodos tradicionais de inspeção

Tradicionalmente, inspeções em espaços confinados são realizadas de forma manual, com trabalhadores entrando em dutos, galerias, tanques e outras estruturas fechadas. Esse processo é reconhecidamente:

- lento, pois depende da mobilidade humana em ambientes restritos;
- perigoso, pela exposição a gases tóxicos, risco de queda, aprisionamento e outras condições adversas;
- altamente dependente da experiência do operador, o que introduz variabilidade nos resultados.

Protocolos como a NR-33 estabelecem requisitos rígidos de segurança (análise de atmosfera, equipamentos de proteção, equipe de resgate, permissão de entrada), o que aumenta o custo operacional, mas não elimina completamente o risco.

O sistema robótico proposto altera essa lógica em vários pontos. Para sintetizar as diferenças entre as abordagens, a Tabela 5 apresenta uma comparação estruturada entre a inspeção manual tradicional e o sistema robótico desenvolvido neste trabalho, considerando critérios fundamentais como segurança, tempo de execução, qualidade da documentação, capacidade analítica e custos associados. Essa comparação permite identificar claramente os benefícios e as limitações de cada abordagem, subsidiando a tomada de decisão sobre a adoção de tecnologias robóticas em contextos específicos de inspeção.

Critério	**Inspeção manual**	**Sistema robótico proposto**
Segurança	Alta exposição humana a riscos	Operador permanece fora do ambiente confinado
Tempo de inspeção	Elevado, depende da mobilidade humana	Potencial redução, robô percorre o trajeto de forma contínua
Documentação	Fotos manuais e relatos subjetivos	Registros padronizados (imagens, telemetria, logs)
Qualidade da análise	Variável, depende da experiência e atenção	IA atua como apoio à identificação de regiões suspeitas
Custo inicial	Baixo (equipamentos simples)	Alto (robô, sensores, processamento embarcado)
Custo operacional	Alto (EPI, equipes, tempo de parada)	Tendência a reduzir após implantação e curva de aprendizado

Tabela 5 – Comparação entre inspeção manual e sistema robótico proposto

Conforme evidenciado na Tabela 5, na configuração atual, o protótipo ainda não elimina a necessidade de um inspetor humano, mas contribui para:

- reduzir a exposição direta a ambientes de risco, ao deslocar o operador para uma zona segura;

- padronizar evidências, por meio de imagens, sobreposições da IA e telemetria registrada;
- organizar melhor os dados de inspeção, facilitando comparações futuras (antes/depois).

Por outro lado, a adoção de um sistema desse tipo traz desafios: investimento inicial mais alto, necessidade de equipe capacitada para configuração, manutenção e operação, além da integração com fluxos de trabalho já estabelecidos nas empresas. Em termos práticos, o protótipo se mostra promissor como complemento às técnicas tradicionais, com potencial de evoluir para substituir etapas mais arriscadas da inspeção manual.

5.3 Comparação com outras tecnologias automatizadas

Além da inspeção manual, diversas soluções automatizadas vêm sendo adotadas para reduzir riscos e aumentar a eficiência, como drones de inspeção, robôs sobre rodas e sistemas fixos de monitoramento. O robô quadrúpede com IA embarcada se posiciona como uma alternativa híbrida entre mobilidade avançada e sensoriamento local de alta resolução.

5.3.1 Drones e robôs de rodas

Drones (aéreos) são extremamente eficientes em ambientes amplos, altos ou de difícil acesso vertical, como fachadas, pontes, fachadas de tanques externos e estruturas industriais ao ar livre. No entanto, em espaços confinados estreitos, com baixa ventilação ou presença intensa de particulados, o voo é limitado por:

- falta de espaço para manobras seguras;
- turbulência gerada pelas próprias hélices;
- risco de colisão com paredes ou estruturas internas;
- restrições de uso por segurança em ambientes com gases inflamáveis.

O robô quadrúpede, por sua vez, se desloca apoiado no solo, o que elimina os problemas associados ao voo em locais muito estreitos. A capacidade de subir desníveis, vencer obstáculos, ajustar a postura e permanecer estático facilita a realização de inspeções detalhadas em locais onde o drone teria dificuldade ou seria proibido.

Já os robôs sobre rodas costumam ser mais simples e econômicos, e funcionam bem em superfícies regulares (pisos lisos, corredores amplos). Em ambientes confinados com pisos irregulares, grades, degraus ou obstáculos, porém, rodas podem ficar presas ou patinar. O sistema quadrúpede leva vantagem justamente nessas condições, pois:

- distribui o peso em quatro apoios com controle dinâmico de postura;
- consegue manter estabilidade em inclinações moderadas;
- adapta melhor a marcha a pequenas irregularidades.

Por outro lado, drones e robôs de rodas ainda podem ser mais vantajosos em distâncias muito longas e cenários menos desafiadores, devido a menor complexidade mecânica e, em alguns casos, maior autonomia energética. Em resumo, o quadrúpede não substitui todas as

plataformas automatizadas, mas preenche um nicho importante: inspeção próxima, em solo, em áreas confinadas e com obstáculos.

5.3.2 Sistemas de câmeras fixas e sensores remotos

Outra abordagem comum é o uso de câmeras fixas, sensores remotos e sistemas de monitoramento permanente instalados na infraestrutura. Essa estratégia é eficiente quando:

- o ambiente é crítico e precisa de monitoramento contínuo (por exemplo, tanques de armazenamento, áreas de processo sensíveis);
- é possível instalar cabos, suportes e proteção mecânica para os sensores;
- o padrão de operação não muda com frequência.

A desvantagem é a baixa flexibilidade espacial: para inspecionar um novo ponto, muitas vezes é necessário instalar novas câmeras ou reposicionar equipamentos, o que implica paradas, obras civis e custos adicionais. Além disso, sistemas fixos tendem a ter campos de visão limitados, podendo deixar "zonas cegas" sem cobertura adequada.

O robô quadrúpede com câmera de profundidade e IA embarcada oferece uma alternativa móvel e reconfigurável:

- pode ser deslocado para diferentes ambientes ou obras, sem necessidade de infraestrutura fixa pré-instalada;
- ajusta o enquadramento conforme a necessidade, aproximando-se de regiões de interesse;
- permite capturar dados sob múltiplos ângulos e distâncias.

Por outro lado, sistemas fixos tendem a ser superiores em monitoramento contínuo 24/7 de pontos específicos, sem envolver mobilidade ou risco mecânico do robô. Uma visão equilibrada é considerar o robô quadrúpede como recurso para inspeções pontuais, missões programadas e campanhas de diagnóstico, enquanto câmeras fixas e sensores remotos permanecem responsáveis pela vigilância permanente de áreas críticas.

Em conjunto, essas comparações reforçam que a solução proposta neste trabalho não busca substituir todas as tecnologias existentes, mas complementá-las, explorando o diferencial da mobilidade quadrúpede aliada à visão computacional embarcada em cenários onde a inspeção manual ou outras plataformas encontram limitações relevantes.

5.3.3 Análise crítica da arquitetura implementada

A arquitetura adotada neste trabalho foi organizada em módulos bem definidos (aquisição de dados, controle e telemetria, processamento de IA e interface de usuário), todos integrados em torno da Jetson Orin NX como unidade central de processamento. Essa separação de responsabilidades se mostrou adequada do ponto de vista de organização do código, manutenção e possibilidade de evolução futura (substituição de componentes sem reescrever todo o sistema).

O uso da NVIDIA Jetson Orin NX como plataforma de processamento embarcado viabilizou a execução local do modelo YOLOv8-s, evitando dependência de serviços em nuvem e permitindo que o robô opere mesmo sem conexão à internet. A integração com o SDK unitree_sdk2 mostrou-se robusta, oferecendo acesso confiável a telemetria e controle do Go2. A câmera Intel RealSense D435i, por sua vez, forneceu imagens RGB adequadas para inspeções de curta distância, típicas de ambientes confinados.

Por outro lado, os testes evidenciaram que a arquitetura, na configuração atual, não alcança desempenho em tempo real pleno para a tarefa de detecção: o pipeline de IA apresentou atrasos e baixa taxa de quadros, tornando a visualização com YOLOv8-s mais adequada como retorno auxiliar do que como visão principal para navegação. Isso indica que, embora o hardware escolhido seja potente, o conjunto modelo + resolução + implementação ainda não está otimizado para o nível de fluidez desejado.

Outra limitação importante é que a arquitetura permanece fortemente dependente de teleoperação humana. A IA atua como camada de apoio visual, mas não há integração direta com módulos de navegação autônoma ou planejamento de trajetória. Do ponto de vista crítico, pode-se dizer que a arquitetura cumpre bem o papel de prova de conceito de inspeção assistida, mas ainda não explora todo o potencial da robótica quadrúpede em termos de autonomia e coordenação entre percepção e movimento.

5.3.4 Limitações observadas

Com base nos resultados obtidos e na experiência prática de uso do protótipo, destacam-se as seguintes limitações:

1. **Desempenho do pipeline de IA em tempo real**
   O modelo YOLOv8-s, na configuração atual, introduz atrasos de alguns segundos no fluxo de vídeo processado, com perda de FPS. Isso reduz a utilidade da visão com IA para navegação dinâmica e limita seu uso a um papel de apoio, principalmente com o robô parado ou em movimentos lentos.
2. **Cobertura e especialização da base de dados**
   O treinamento foi feito sobre um dataset genérico de corrosão (*Rust Detection Dataset* (Roboflow Universe, 2025)), o que é adequado para uma fase inicial, mas não garante cobertura completa dos tipos de defeitos encontrados em estruturas reais. Isso se reflete em falsos positivos em manchas não corrosivas e em falhas na detecção de regiões pequenas ou pouco contrastadas.
3. **Dependência de teleoperação manual**
   A navegação e o posicionamento da câmera ainda dependem integralmente do operador. A IA não influencia diretamente as decisões de trajetória, nem há módulos de planejamento automático para revisitar ou explorar áreas identificadas como suspeitas.
4. **Sensibilidade a condições de cena e enquadramento**

Embora não tenham sido realizados testes exaustivos em campo, mesmo nos ensaios de bancada já se observou que o desempenho da detecção varia com o enquadramento, a proximidade da superfície e a qualidade visual da área de interesse. Variações de iluminação, reflexos e textura podem prejudicar a confiabilidade do modelo.

5. **Ausência de validação sistemática em cenário real**
   Os testes foram realizados em condições controladas de bancada, com chapas e peças metálicas. Ainda não há uma avaliação sistemática em dutos, galerias ou estruturas industriais em operação, o que limita as conclusões quanto à robustez do sistema em ambiente de produção.

Essas limitações não invalidam os resultados, mas delimitam com clareza o escopo atual do protótipo e orientam os próximos passos de evolução da solução.

5.3.5 Propostas de melhoria

À luz da análise crítica e das limitações observadas, é possível apontar uma série de melhorias graduais em três frentes principais: hardware, software/IA e operação/usabilidade.

Hardware

- **Sensores adicionais (térmicos e de gases)**
  A integração de sensores térmicos e de gases (por exemplo, CO_2, CH_4, O_2, H_2S) ampliaria o escopo do sistema para além da inspeção visual, permitindo monitorar condições atmosféricas e riscos associados à presença de gases perigosos em espaços confinados.
- **Câmeras com obturador global e óptica ajustável**
  A adoção de câmeras com obturador global pode reduzir efeitos de motion blur durante o deslocamento do robô, melhorando a nitidez das imagens em movimento. Lentes varifocais ou conjuntos ópticos com diferentes campos de visão também podem ser avaliados para otimizar a inspeção a curta e média distância.
- **Conectividade avançada (5G ou rede privada de alta capacidade)**
  A integração futura com módulos de comunicação 5G ou redes privadas de alta capacidade permitiria transmitir vídeo e dados de forma mais confiável em ambientes industriais complexos, viabilizando supervisão remota em tempo quase real, mesmo quando o robô opera em áreas mais extensas.

Software e IA

- **Otimização e escolha de modelos mais leves**
  Testar variantes mais leves do YOLO (como YOLOv8-n) e aplicar técnicas de quantização, pruning e TensorRT pode reduzir significativamente a latência de inferência, aproximando o sistema de um comportamento realmente “tempo real”.

- **Treinamento contínuo com dados de campo**
  Incorporar imagens capturadas pelo próprio robô em ambientes reais, com anotação progressiva, permitiria realizar fine-tuning contínuo do modelo. Essa abordagem de "learning on the edge" melhora a aderência da IA ao domínio específico de uso.
- **Exploração de arquiteturas complementares**
  Em etapas futuras, pode ser interessante avaliar arquiteturas de segmentação (como U-Net ou Mask R-CNN) e/ou modelos baseados em Transformers para detecção e análise contextual de anomalias, combinando detecção de objetos com segmentação de regiões de fissura/corrosão.
- **Uso explícito da informação de profundidade**
  A profundidade fornecida pela RealSense ainda é utilizada de forma indireta. Desenvolver um pipeline que integre diretamente mapas de profundidade (por exemplo, filtrando regiões em relevo anormal ou destacando descontinuidades na superfície) pode melhorar a robustez da detecção, diferenciando melhor manchas superficiais de defeitos estruturais.

Operação e usabilidade

- **Melhorias na interface e relatórios automáticos**
  A interface atual já agrega vídeo, métricas e controle, mas ainda não gera relatórios estruturados. Uma evolução natural é a criação de um painel web para relatórios automáticos, com organização das imagens, sobreposições da IA, telemetria e comentários do operador, facilitando a comparação temporal (antes/depois).
- **Telemetria remota e alertas**
  A adoção de um módulo de telemetria remota com alertas automáticos de falhas (queda de bateria, perda de comunicação, temperatura elevada) aumentaria a segurança operacional e facilitaria o acompanhamento de missões à distância.
- **Recursos avançados de interação**
  Futuramente, podem ser avaliadas formas alternativas de interação com o sistema, como comandos por voz ou atalhos simplificados na interface para operadores em ambiente hostil (uso de luvas, visores, etc.). Embora não sejam essenciais na fase atual, esses recursos podem melhorar a usabilidade em contextos industriais mais exigentes.

De forma geral, as propostas de melhoria indicam que o sistema atual funciona como uma plataforma de base: já entrega uma prova de conceito funcional, mas foi projetado de forma modular justamente para permitir incrementos graduais de sensoriamento, inteligência e automação à medida que novos requisitos surgirem e mais dados de campo forem incorporados ao projeto.

5.3.6 Relevância acadêmica e industrial

Do ponto de vista acadêmico, o trabalho se destaca por mostrar, de forma concreta, como integrar três linhas de pesquisa que muitas vezes aparecem separadas na literatura: robótica móvel quadrúpede, visão computacional de profundidade e IA embarcada para detecção de anomalias. A arquitetura proposta com Go2, RealSense D435i, Jetson Orin NX e um pipeline de IA executando localmente forma um arranjo replicável em laboratórios de engenharia e computação que desejem explorar inspeção inteligente em ambientes complexos.

Além disso, o projeto documenta:

- boas práticas de integração mecânica e elétrica de payloads em robôs quadrúpedes;
- um fluxo completo de treinamento e implantação de modelo YOLOv8-s em ambiente embarcado;
- um protótipo de interface web voltada a teleoperação e visualização de detecções.

Esses elementos podem servir como base para disciplinas de robótica, visão computacional, sistemas embarcados e TCCs futuros, tanto para estender a solução (por exemplo, adicionando navegação autônoma) quanto para comparar diferentes arquiteturas de IA.

Sob a ótica industrial, o sistema aborda um problema diretamente ligado à segurança do trabalho: inspeções em espaços confinados, reguladas por normas como a NR-33. Ao deslocar o operador para fora da zona de risco e utilizar o robô como primeiro agente de inspeção, o projeto aponta um caminho concreto para:

- reduzir a exposição humana a atmosferas perigosas, confinamento físico e outros riscos típicos;
- padronizar evidências por meio de imagens, sobreposições da IA e registros de telemetria;
- apoiar iniciativas de manutenção preditiva, facilitando a comparação temporal entre inspeções.

Embora o protótipo ainda esteja em estágio experimental e focado em ensaios de bancada, ele demonstra o potencial de soluções desse tipo para empresas de construção, saneamento, mineração, óleo e gás, entre outras, abrindo espaço para parcerias universidade-indústria e para a evolução rumo a produtos e serviços comercialmente viáveis.

5.3.7 Considerações finais da discussão

A discussão dos resultados permite afirmar que o sistema desenvolvido atingiu seu papel de prova de conceito: mostrou ser possível acoplar um robô quadrúpede a uma cadeia de visão de profundidade e IA embarcada, construir um pipeline funcional de aquisição, processamento e visualização, além de conduzir inspeções assistidas em condições controladas, com o modelo de IA atuando como apoio ao operador.

Ao mesmo tempo, os testes evidenciaram limitações importantes (em especial o gargalo de desempenho do modelo YOLOv8-s em tempo real e a dependência de um dataset genérico de

corrosão) que impedem, por enquanto, a aplicação direta do protótipo em rotinas industriais sem uma etapa adicional de amadurecimento tecnológico. Em vez de fragilizar o trabalho, essas limitações ajudam a delimitar com clareza o nível de prontidão tecnológica alcançado e apontam caminhos concretos de evolução (otimização do modelo, coleta de dados em campo, integração com navegação autônoma).

De forma geral, a discussão mostra que:

- a arquitetura modular adotada é coerente e facilita incrementos graduais;
- o ganho de segurança potencial é evidente, ao retirar o trabalhador de dentro do espaço confinado;
- a IA, mesmo com desempenho moderado, já contribui para direcionar a atenção do inspetor humano.

Assim, o capítulo reforça a ideia de que o trabalho não se encerra em um produto acabado, mas em um framework prático e pesquisável para inspeções técnicas com robôs quadrúpedes, sobre o qual novas camadas de inteligência, autonomia e sensoriamento podem ser adicionadas nos próximos ciclos de pesquisa e desenvolvimento.

6 Conclusão

Este capítulo final consolida os principais achados da pesquisa, retomando o problema investigado, os objetivos estabelecidos e a hipótese formulada, confrontando-os com os resultados obtidos ao longo do desenvolvimento do sistema de inspeção. São apresentadas as contribuições técnicas e científicas do trabalho, as limitações reconhecidas e as recomendações para trabalhos futuros. A conclusão busca sintetizar o conhecimento gerado, destacando o nível de prontidão tecnológica alcançado pelo protótipo e seu potencial de evolução para aplicações industriais reais em espaços confinados.

6.1 Síntese do estudo

Este trabalho desenvolveu e avaliou um sistema integrado de inspeção para espaços confinados, composto por um robô quadrúpede Unitree Go2, uma câmera de profundidade Intel RealSense D435i e um módulo de processamento NVIDIA Jetson Orin NX 16 GB. A solução proposta envolveu a integração mecânica e elétrica do payload ao robô, a configuração de uma camada de software embarcada para aquisição sensorial e controle, a implementação de uma interface web de teleoperação e a implantação de um modelo de IA baseado em YOLOv8-s para detecção de corrosão em superfícies metálicas.

A pesquisa seguiu uma abordagem incremental: primeiro, foram definidos os requisitos de hardware e software; em seguida, realizou-se a montagem e integração do protótipo; depois, configurou-se o ambiente de processamento na Jetson e o SDK do robô; e, por fim, foram conduzidos ensaios de bancada com chapas e peças metálicas contendo corrosão, a fim de validar o pipeline de visão embarcado e a interface de operação.

Os resultados mostraram que o sistema é tecnicamente viável: o robô foi capaz de operar com o payload instalado, a comunicação entre Go2, Jetson e câmera funcionou de forma consistente e o modelo de IA conseguiu destacar regiões com padrão visual de corrosão nas cenas de teste. Ao mesmo tempo, ficaram evidentes limitações importantes, especialmente o gargalo de desempenho do YOLOv8-s em tempo real e a dependência de uma base de dados ainda genérica para o domínio alvo.

Em síntese, o estudo entrega um protótipo funcional de inspeção assistida por IA, documenta um arranjo de hardware e software replicável e estabelece uma base concreta para evoluções futuras em direção a aplicações industriais mais maduras.

6.2 Retomada do problema e objetivos

O problema de pesquisa que orientou este trabalho pode ser resumido da seguinte forma:

Como integrar um robô quadrúpede com visão de profundidade e IA embarcada para apoiar a detecção de anomalias estruturais em espaços confinados, reduzindo a exposição humana ao risco e aumentando a padronização das evidências coletadas?

A partir dessa questão central, foram definidos os seguintes objetivos gerais e específicos:

Objetivo geral

Construir e validar um sistema robótico capaz de capturar, processar e analisar imagens localmente, com suporte de IA, para auxiliar inspeções em espaços confinados.

Objetivos específicos (resumidos)

a) realizar a integração mecânica e eletrônica entre o robô Go2, a Jetson Orin NX e a câmera RealSense D435i, respeitando as diretrizes de payload do fabricante;
b) implementar a camada de software de base (SDK da Unitree, aquisição de dados, telemetria, sincronização e serviços na Jetson);
c) desenvolver uma interface de teleoperação e visualização, permitindo controle do robô, monitoramento de métricas e visualização de vídeo RGB/Depth;
d) selecionar e organizar uma base de dados de imagens rotuladas adequada ao problema de corrosão;
e) treinar e implantar um modelo de IA (YOLOv8-s) otimizado para execução embarcada;
f) realizar ensaios experimentais de bancada, avaliando a integração do pipeline e observando o comportamento do modelo em situações próximas ao uso real;
g) documentar os procedimentos de instalação, atualização e operação do sistema, em consonância com a documentação oficial do fabricante.

No decorrer do trabalho, todos esses objetivos foram alcançados em algum grau: o protótipo físico foi montado e testado, o software embarcado foi implementado, a interface foi utilizada nos ensaios, o modelo de IA foi treinado com o *Rust Detection Dataset* (Roboflow Universe, 2025) e implantado na Jetson, e os testes de bancada permitiram avaliar o fluxo completo de captura $\rightarrow$ IA $\rightarrow$ visualização.

Entretanto, a etapa de validação em ambiente real de campo permaneceu fora do escopo desta fase do projeto, restringindo-se a cenários controlados de bancada. Assim, os objetivos foram cumpridos no contexto de uma prova de conceito experimental, o que é compatível com a natureza de um Trabalho de Conclusão de Curso, mas deixa espaço aberto para fases posteriores de validação industrial.

6.3 Verificação da hipótese

A hipótese inicialmente formulada pode ser expressa, em termos gerais, como:

A integração de um robô quadrúpede com visão de profundidade e IA embarcada permite detectar anomalias estruturais em espaços confinados com baixa latência e maior segurança operacional, configurando uma alternativa viável às inspeções exclusivamente manuais.

Com base nos resultados obtidos, essa hipótese pode ser considerada em grande parte confirmada, porém com ressalvas importantes:

- **Do ponto de vista da segurança**, a hipótese se sustenta: o sistema permite que o robô seja o primeiro agente a entrar em áreas confinadas, deslocando o operador para uma zona segura e reduzindo a exposição direta a riscos físicos e ambientais.
- **Em relação à detecção de anomalias**, o modelo YOLOv8-s demonstrou capacidade de identificar regiões com padrão visual de corrosão nos ensaios de bancada, atuando como uma ferramenta de apoio que destaca áreas suspeitas e organiza evidências visuais. No entanto, a presença de falsos positivos e de falhas na detecção de áreas pequenas indica que a detecção ainda não é suficientemente robusta para substituir o julgamento humano, mas sim para complementá-lo.
- **Quanto à latência**, a hipótese precisa ser qualificada: embora o processamento seja feito localmente na Jetson (sem envio para a nuvem), o pipeline com YOLOv8-s apresentou atrasos perceptíveis e baixa taxa de quadros, o que impede, nesta fase, classificá-lo como "baixa latência" em uso contínuo como visão principal. A operação prática adotou a imagem RGB sem IA como referência principal para teleoperação, com a IA sendo usada como overlay auxiliar.

Em termos conclusivos, pode-se dizer que a hipótese foi parcialmente confirmada:

- confirmou-se a viabilidade técnica da integração robô + visão de profundidade + IA embarcada, bem como o potencial de aumento de segurança e melhoria da rastreabilidade das inspeções;
- mas ficou claro que ainda são necessárias otimizações de desempenho e aperfeiçoamentos do modelo de IA e da base de dados para que o sistema atinja plenamente o nível de latência, robustez e autonomia sugerido na formulação original da hipótese.

Essa constatação não diminui a relevância do trabalho; ao contrário, delimita com precisão o nível de prontidão tecnológica alcançado e indica, de maneira objetiva, os passos seguintes para a evolução da solução em projetos futuros.

6.4 Contribuições do trabalho

6.4.1 Contribuições técnicas e científicas

- Arranjo replicável de hardware e software para inspeção com robô quadrúpede, documentando boas práticas de montagem do payload (fixação, centro de gravidade, roteamento de cabos, alimentação) e procedimentos básicos de atualização de firmware e software em conformidade com a documentação da Unitree.

- Pipeline de visão embarcado na Jetson Orin NX, integrando câmera de profundidade, modelo de detecção (YOLOv8-s) e interface web. Ainda que não atinja, nesta fase, latência ideal para uso como visão principal em movimento, o pipeline demonstra de forma prática como realizar detecção embarcada e registro estruturado de evidências visuais.
- Interface de operação integrada, combinando teleoperação do Go2, visualização multimodal (fluxo RGB "limpo" e fluxo com IA), métricas básicas de telemetria e mecanismos simples de registro de imagens, o que contribui para reduzir subjetividade na inspeção e facilitar o pós-processamento manual.
- Protocolo experimental em ensaios de bancada, com foco em superfícies metálicas contendo corrosão, que permite observar o comportamento do modelo quanto a acertos, falsos positivos e limitações de enquadramento/iluminação, servindo como base para futuros protocolos mais completos em ambientes industriais reais.

6.4.2 Relevância prática (engenharia/indústria)

- Mitigação de risco humano, ao colocar o robô como primeiro agente dentro do espaço confinado e manter o operador em área segura, alinhado aos princípios de segurança estabelecidos por normas como a NR-33.
- Potencial de aumento da eficiência e da qualidade de documentação das inspeções, por meio de imagens padronizadas, sobreposições da IA e registros de telemetria, permitindo comparações temporais (antes/depois) e maior rastreabilidade das intervenções.
- Base tecnológica para iniciativas de manutenção preditiva, uma vez que o sistema organiza dados visuais e operacionais em formato adequado para futura integração com plataformas de gestão de ativos, mesmo ainda estando em estágio de protótipo experimental.

6.5 Limitações reconhecidas

As principais limitações identificadas ao longo do trabalho podem ser resumidas em:

- **Desempenho em tempo real do modelo de IA**: O YOLOv8-s, na configuração atual, introduz atraso perceptível e baixa taxa de FPS no fluxo de vídeo processado, o que restringe seu uso a um papel de apoio visual, especialmente quando o robô está parado ou com pouco movimento.
- **Sensibilidade a iluminação e superfícies metálicas**: A câmera Intel RealSense D435i apresenta degradação na qualidade do mapa de profundidade em superfícies metálicas reflexivas ou sob iluminação pouco favorável. Ajustes de exposição e filtros de ruído ajudam, mas não eliminam totalmente o problema.
- **Dependência de um dataset genérico**: O modelo foi treinado sobre um dataset genérico de corrosão, que não contempla todas as texturas e condições presentes em estruturas

reais. Isso resulta em falsos positivos/negativos, especialmente em casos de corrosão incipiente ou manchas visuais ambíguas.

- **Teleoperação predominante e ausência de navegação autônoma**: A navegação do robô ainda depende integralmente do operador humano; a IA não influencia diretamente o planejamento de trajetória. Isso exige operadores treinados e limita o grau de automação alcançado nesta fase.
- **Autonomia energética limitada para missões longas**: Embora suficiente para ensaios de bancada e missões curtas, a autonomia da bateria do robô e do payload embarcado impõe restrições a inspeções extensas, demandando planejamento cuidadoso de recarga ou uso de baterias adicionais em aplicações reais.

6.6 Recomendações e trabalhos futuros

6.6.1 Evolução de hardware e sensoriamento

- **Integração de sensores adicionais (térmicos e de gases)** Acrescentar sensores de temperatura e de gases (como CH_4, O_2, H_2S, entre outros) para ampliar a análise situacional e gerar alertas de segurança em tempo real, complementando a inspeção visual.
- **Avaliação de câmeras e ópticas alternativas** Testar câmeras com obturador global e lentes varifocais para reduzir motion blur e melhorar a qualidade de imagens em diferentes distâncias e condições de iluminação.
- **Conectividade avançada e energia** Explorar o uso de 5G ou redes privadas de alta capacidade (Private LTE) para streaming de vídeo e dados com menor latência; avaliar o uso de baterias extras e bases de recarga para suportar missões prolongadas em campo.

6.6.2 Evolução de software e IA

- **Navegação autônoma baseada em SLAM multimodal** Integrar LiDAR 4D, IMU e visão estéreo em um pipeline de SLAM multimodal, permitindo planejamento de trajetória, mapeamento do ambiente e desvio automático de obstáculos em espaços confinados.
- **Arquiteturas híbridas de detecção e segmentação** Investigar combinações entre modelos de detecção (YOLO) e segmentação (como U-Net ou Mask R-CNN) para aumentar a sensibilidade a fissuras finas e permitir estimar extensão/área de regiões corroídas.
- **Aprendizado contínuo e semi-supervisionado** Implementar estratégias de continual learning com curadoria semiautomática de novos dados capturados em campo, bem como técnicas semi-supervisionadas e de auto-rotulagem assistida para aproveitar imagens não rotuladas.
- **Uso explícito de profundidade e interpretabilidade** Incorporar a informação de profundidade de forma direta na decisão do modelo (por exemplo, destacando descontinuidades

de superfície) e explorar métodos de explicabilidade, como Grad-CAM ou mapas de saliência, para tornar mais transparentes os motivos das detecções de fissura/corrosão.

6.6.3 Operação, governança e dados

- **Procedimentos Operacionais Padrão (POP)** Formalizar POPs para pré-missão, missão e pós-missão, incluindo checklists de segurança, verificação de firmware, coleta e descarte de dados, e critérios de aceitação de inspeções.
- **Governança de dados e modelos** Estabelecer práticas de versionamento de modelos, controle de qualidade de rótulos, trilhas de auditoria e políticas de retenção/privacidade para os dados coletados, garantindo rastreabilidade e conformidade regulatória.
- **Capacitação de equipe** Desenvolver programas de treinamento para operadores, engenheiros e equipe de manutenção, abordando uso do robô, sensores, pipeline de IA, redes e procedimentos de segurança específicos para espaços confinados.

6.6.4 Implicações éticas e de segurança

Mesmo com a introdução de um robô quadrúpede como agente principal de inspeção, os requisitos de segurança para espaços confinados (como os previstos na NR-33) permanecem válidos: entrada controlada, monitoramento atmosférico, equipe de resgate de prontidão e meios de comunicação redundantes continuam essenciais. O sistema proposto deve ser visto como uma camada adicional de proteção, e não como substituto integral desses protocolos.

Do ponto de vista de segurança cibernética, a conexão remota ao robô e à Jetson exige cuidados com autenticação, criptografia, segmentação de rede e registro de acessos, para evitar uso indevido, manipulação de dados ou interferência em missões de inspeção.

Em termos de transparência e responsabilidade, é importante que os relatórios gerados a partir do sistema preservem evidências verificáveis (imagens originais, overlays da IA, metadados de tempo e posição), permitindo auditorias técnicas e garantindo que decisões de manutenção ou interdição de áreas não se baseiem em resultados opacos ou impossíveis de reconstituir.

6.6.5 Considerações finais

O trabalho demonstrou que a combinação de robótica quadrúpede, visão de profundidade e IA embarcada constitui uma abordagem viável e promissora para apoio a inspeções em espaços confinados. A solução proposta aumenta o potencial de segurança operacional ao afastar o trabalhador da zona de risco, padroniza a coleta de evidências visuais e cria uma base objetiva para análise de anomalias estruturais.

Ao mesmo tempo, a conclusão destaca que o sistema ainda se encontra em estágio de prova de conceito: o gargalo de desempenho do modelo de IA, a dependência de um dataset genérico

e a ausência de navegação autônoma plena são desafios que precisam ser enfrentados antes que a solução possa ser considerada pronta para adoção em larga escala na indústria.

Apesar dessas limitações, o projeto estabelece um patamar tecnológico sólido sobre o qual futuras evoluções podem ser construídas, incluindo sensoriamento multimodal, navegação autônoma, aprendizado contínuo e integração com plataformas de gestão de ativos. Em termos práticos, o TCC entrega um framework prático e cientificamente fundamentado para inspeções técnicas com robôs quadrúpedes, capaz de orientar novos desenvolvimentos, apoiar decisões de engenharia e contribuir para a redução de riscos em ambientes confinados.

Nota sobre estrutura acadêmica:

Esta conclusão foi elaborada em consonância com a estrutura clássica de monografias adotada pela instituição, contemplando síntese do estudo, retomada do problema e objetivos, verificação de hipótese, contribuições, limitações e propostas de trabalhos futuros.

Referências

ABNT. *ABNT NBR 6023:2018 – Informação e documentação – Referências – Elaboração*. Rio de Janeiro, 2018.

BECOY, A. J. et al. Autonomous navigation of quadrupeds using coverage path planning with morphological skeleton map. *arXiv preprint arXiv:2504.17880*, June 2025.

BICK, A. *Quadrupedal Gait on Oscillating Surfaces*. Dissertação (Master thesis) — Technical University of Darmstadt, Darmstadt, Germany, May 2025. Department of Computer Science.

Bureau of Labor Statistics. *Census of Fatal Occupational Injuries: Confined Spaces*. 2021. Disponível em: <https://www.bls.gov/>. Acesso em: nov. 2025.

CARNETTI, W. C.; CALVO, T. C. M. Análise do sistema de gestão de riscos em espaços confinados implantado na usina hidrelétrica de mascarenhas. *Destarte – Revista de Divulgação Científica da Estácio*, v. 9, n. 1, p. 99–118, 2020. Acesso em: nov. 2025.

CHOWDHURY, I. H. Bachelor's Degree, *Integrated Monitoring and Control Interface for Unitree Go2 Robot*. Finland: [s.n.], 2025. Technology, Communication and Transport – Degree Programme in Information Technology.

DENG, B. et al. Learning to recover: Dynamic reward shaping with wheel-leg coordination for fallen robots. *arXiv preprint arXiv:2506.05516*, October 2025.

DING, J. et al. Towards dynamic quadrupedal gaits: A symmetry-guided rl hierarchy enables free gait transitions at varying speeds. *arXiv preprint arXiv:2510.10455*, October 2025.

FUNDACENTRO. *Relatório de Acidentes em Espaços Confinados*. São Paulo, 2022.

GOMES, A.; SEO, H. Ground robots for confined space inspection: Challenges and opportunities. *Robotics and Autonomous Systems*, 2022.

Intel Corporation. *Intel RealSense Depth Camera D435i – Product Specifications*. Santa Clara, CA: [s.n.], 2023. Disponível em: <https://www.intelrealsense.com/depth-camera-d435i/>. Acesso em: nov. 2025.

Ministério do Trabalho e Emprego. *NR-33 – Segurança e Saúde nos Trabalhos em Espaços Confinados*. Brasília, 2006. Norma Regulamentadora.

Ministério do Trabalho e Previdência. *Relatório de Análise de Impacto Regulatório (AIR) da NR-33*. Brasília, 2021.

NOGUEIRA, A. et al. Challenges in using drones for confined space inspection. *Robotics and Autonomous Systems*, 2020. DOI: 10.1016/j.robot.2020.xxxxx.

NVIDIA Corporation. *Jetson Orin NX Series – Data Sheet*. Santa Clara, CA, 2023. Acesso em: nov. 2025.

OpenCV Team. *OpenCV – Open Source Computer Vision Library*. 2024. Versão 4.8. Disponível em: <https://opencv.org/>. Acesso em: nov. 2025.

PyTorch Team. *PyTorch – An open source machine learning framework*. 2024. Versão 2.1. Disponível em: <https://pytorch.org/>. Acesso em: nov. 2025.

Roboflow Universe. *Rust Detection Dataset*. 2025. Projeto rust-detection-38s6e, versão 1. Licença: CC BY 4.0. Disponível em: <https://universe.roboflow.com/>. Acesso em: nov. 2025.

Ultralytics. *Ultralytics YOLOv8 Documentation*. 2025. Disponível em: <https://docs.ultralytics.com/>. Acesso em: nov. 2025.

Unitree Robotics. *Unitree Go2 – Developer SDK Concepts, Payload and Module Update Guides*. Hangzhou, China, 2025. Acesso em: nov. 2025.

Unitree Robotics. *Unitree SDK2 – Robot SDK Version 2*. 2025. Repositório GitHub. Disponível em: <https://github.com/unitreerobotics/unitree_sdk2>. Acesso em: nov. 2025.

Unitree Robotics. *Unitree Technology Document Center*. 2025. Disponível em: <https://support.unitree.com/home/en/developer>. Acesso em: nov. 2025.

Universidade Católica do Salvador. *Regulamento TCC – Resolução nº 10/2022*. Salvador, 2022.

U.S. Bureau of Labor Statistics. *Fatal occupational injuries involving confined spaces, 2011–2018*. [S.l.], 2020. Acesso em: nov. 2025.

WILHELM, S. *Autonomous Navigation and Real-Time 3D Reconstruction of Interior Spaces Using a Quadruped Robot*. Dissertação (Master of Science Thesis) — University of Central Florida, Orlando, Florida, 2025. School of Modeling, Simulation and Training.

ZHOU, Y. et al. Identification of surface cracks using yolo and resnet architectures. *Journal of Structural Health Monitoring*, v. 22, n. 3, p. 415–427, 2025.

Apêndices

APÊNDICE A – Especificações Técnicas Detalhadas

A.1 Especificações do Robô Unitree Go2

Característica	**Especificação**
Peso	Aproximadamente 15 kg
Capacidade de carga	7 a 10 kg
Velocidade máxima	>3 m/s
Tensão de operação	28 V a 33,6 V
Potência máxima	3000 W
Campo de visão LiDAR	360° × 90°
Distância mínima LiDAR	0,05 m
Câmera frontal	HD 1280×720, 120° FOV
Conectividade	Wi-Fi 6, Bluetooth 5.2, 4G eSIM

Tabela 6 – Especificações técnicas do Unitree Go2 EDU

A.2 Especificações da Câmera Intel RealSense D435i

Característica	**Especificação**
Faixa operacional	0,3 m a 3 m
Resolução de profundidade	Até 1280×720 @ 30 fps
Campo de visão	87° (H) × 58° (V)
IMU	Acelerômetro + giroscópio (6 DoF)
Peso	Aproximadamente 72 g
Interface	USB 3.1
Tecnologia	Estéreo ativo com projeção IR

Tabela 7 – Especificações técnicas da Intel RealSense D435i

A.3 Especificações da NVIDIA Jetson Orin NX 16 GB

Característica	**Especificação**
GPU	Ampere com 1024 CUDA Cores + 32 Tensor Cores
CPU	ARM Cortex-A78AE com 8 núcleos
Memória	16 GB LPDDR5 (128 bits)
Desempenho	Até 100 TOPS (INT8 sparse)
Consumo de energia	10–25 W (configurável)
Periféricos	Até 8 câmeras CSI ou 4 portas USB
Sistema operacional	JetPack (Ubuntu 20.04/22.04)

Tabela 8 – Especificações técnicas da NVIDIA Jetson Orin NX

APÊNDICE B – Procedimentos de Instalação e Configuração

B.1 Checklist de Pré-Missão

1. Verificar nível de bateria do robô (mínimo 50%)
2. Verificar conexões físicas (câmera, Jetson, cabos)
3. Testar comunicação Ethernet: `ping 192.168.123.161`
4. Verificar captura da câmera RealSense
5. Testar modelo de IA (executar inferência de teste)
6. Verificar acesso à interface web
7. Validar espaço de armazenamento disponível
8. Confirmar iluminação adequada no ambiente

B.2 Comandos Básicos de Operação

B.2.1 Inicialização do Sistema

```
# Conectar via SSH à Jetson
ssh jetson@192.168.123.222

# Ativar ambiente virtual
source ~/venv/bin/activate

# Iniciar serviços
sudo systemctl start go2-control.service
sudo systemctl start camera-acquisition.service
sudo systemctl start ai-inference.service
sudo systemctl start web-interface.service
```

B.2.2 Captura de Dados

```
# Salvar frame RGB
python3 capture_frame.py --output /data/frames/

# Salvar mapa de profundidade
python3 capture_depth.py --output /data/depth/
```

```
# Iniciar gravação de nuvem de pontos
python3 lidar_capture.py --start --duration 60
```