Dhara Ianne Pamplona de Matos

Sistema de Visão Computacional com ROS para Identificação e Rastreamento de Objetos em Movimento em uma Esteira Industrial

Campina Grande, PB

2024

Dhara Ianne Pamplona de Matos

Sistema de Visão Computacional com ROS para Identificação e Rastreamento de Objetos em Movimento em uma Esteira Industrial

Trabalho de Conclusão de Curso (TCC) submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia Elétrica.

Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Departamento de Engenharia Elétrica

Orientador: Angelo Perkusich

Campina Grande, PB

2024

Dhara Ianne Pamplona de Matos

Sistema de Visão Computacional com ROS para Identificação e Rastreamento de Objetos em Movimento em uma Esteira Industrial

Trabalho de Conclusão de Curso (TCC) submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia Elétrica.

Trabalho aprovado. Campina Grande, PB, 5 de novembro de 2024

Angelo Perkusich
Orientador

Kyller Costa Gorgônio
Convidado

Campina Grande, PB
2024

Agradeço aos meus pais pelo apoio incondicional em todos os momentos difíceis da minha trajetória acadêmica. Este trabalho é dedicado a eles.

Agradecimentos

Gostaria de expressar, em primeiro lugar, minha profunda gratidão aos meus pais, Adriana e Itamar, que foram meu porto seguro durante toda essa jornada acadêmica. Pelo amor, apoio incondicional e dedicação ao longo de todos esses anos, sou eternamente grata. Sem vocês, nada disso seria possível. Agradeço também ao meu querido irmão Dhiego, que nunca deixou faltar motivação e esteve sempre ao meu lado, incentivando-me a seguir em frente.

Um agradecimento especial à Marina e à Sabrina, companheiras que dividiram comigo muitas noites em claro para desenvolver este trabalho. Juntas, enfrentamos os desafios com muita dedicação, e essa parceria foi indispensável. Também sou grata aos meus colegas de curso Andrey, Débora, Leiry e Lucas, que tornaram essa jornada mais leve e cheia de aprendizado. Sou igualmente grata à Melissa e à Karen, colegas do laboratório Embedded, que contribuíram com seu apoio e colaboração ao longo do percurso.

Agradeço, com carinho e admiração, ao meu namorado Jan Paulo, que esteve ao meu lado nos momentos mais difíceis. Ele foi minha força e incentivo nos períodos de fragilidade, sempre acreditando em mim e me ajudando a seguir em frente.

Sou profundamente grata aos meus amigos Gleisson, Gabriel, Nattan, Haenell e Junior, que passaram várias noites me fazendo companhia enquanto eu realizava tarefas acadêmicas. A presença de vocês e os momentos de descontração que proporcionaram foram essenciais para que eu mantivesse a motivação e a energia ao longo desse caminho.

A minha amiga de longa data, Sandy, que mesmo à distância sempre esteve presente em minha vida. Sua amizade constante, mesmo em outra cidade, e sua confiança em meu potencial me motivou a alcançar este objetivo.

A cada uma dessas pessoas, minha eterna gratidão, pois cada uma delas teve um papel importante para que esta conquista fosse possível. Suas presenças e incentivos fizeram toda a diferença nesta etapa tão marcante de minha vida.

“ O próprio Senhor irá à sua frente e estará com você; ele nunca o deixará, nunca o abandonará. Não tenha medo! Não se desanime! ”

Deuteronômio 31:8

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ROS	*Robot Operating System*
BTs	*Behavior Trees*
UFCG	Universidade Federal de Campina Grande
CEEI	Centro de Engenharia Elétrica e Informática

Resumo

Desenvolvimento de um sistema de visão computacional integrado com o Robot Operating System para a identificação e rastreamento de objetos em movimento em uma esteira industrial. Utilizando câmeras e sensores, o sistema detecta e rastreia objetos através de marcadores fiduciais, como ArUco. A implementação foi validada em um ambiente controlado, empregando metodologias de calibração e integração de componentes, permitindo uma automação robusta e flexível para linhas de produção industriais. Os resultados indicam que o sistema é eficaz para o monitoramento e a classificação de objetos em tempo real, contribuindo para a eficiência e a adaptabilidade das operações industriais.

Palavras chave: visão computacional; ROS; ArUco; manipulação robótica; esteira industrial; automação industrial; calibração de câmera.

Abstract

Development of a computer vision system integrated with the Robot Operating System for identifying and tracking moving objects on an industrial conveyor belt. Using cameras and sensors, the system detects and tracks objects through fiducial markers, such as ArUco. The implementation was validated in a controlled environment, employing calibration methodologies and component integration, enabling robust and flexible automation for industrial production lines. The results indicate that the system is effective for real-time monitoring and classification of objects, contributing to the efficiency and adaptability of industrial operations.

Keywords: computer vision; ROS; ArUco; robotic manipulation; industrial conveyor belt; industrial automation; camera calibration.

Sumário

1 Introdução

A Indústria 4.0 trouxe inovações significativas para os processos industriais. Com o crescimento das indústrias, houve um grande avanço das tecnologias, e, entre elas, a visão computacional ganhou destaque. Essa tecnologia permite que máquinas e sistemas interpretem informações visuais, viabilizando a automação de tarefas como monitoramento, identificação e rastreamento de objetos. Com isso, a visão computacional se tornou um elemento essencial para o aumento da eficiência, flexibilidade e precisão em linhas de produção modernas.

1.1 Justificativa

A transformação digital promovida pela Indústria 4.0 tem impulsionado a automação e a integração de tecnologias avançadas nos processos industriais, visando aumentar a eficiência, flexibilidade e conectividade nas linhas de produção. Um dos principais pilares dessa transformação é o uso de sistemas de visão computacional, que permitem monitorar, identificar e rastrear objetos em tempo real, facilitando o controle de qualidade e a tomada de decisões de maneira descentralizada e autônoma.

De acordo com (GHOSON et al., 2023), o uso de sensores de visão em robôs tem se mostrado essencial para o monitoramento e controle de máquinas em ambientes de manufatura. Sistemas de visão inteligente, como o proposto neste trabalho, podem ser aplicados para a identificação de objetos em movimento em uma esteira, integrando-se a outros componentes da linha de produção para aumentar a autonomia e a flexibilidade das operações. Além disso, o uso de câmeras permite o monitoramento contínuo do fluxo de produção, possibilitando ações corretivas em tempo real.

A implementação de sistemas ciberfísicos e de Internet das Coisas, conforme discutido em (CAñAS et al., 2021), é fundamental para a Indústria 4.0, pois permite a comunicação e interoperabilidade entre dispositivos, facilitando a coordenação de tarefas automatizadas. Esse contexto justifica a adoção de um sistema de visão computacional baseado no ROS, que é altamente modular e escalável, permitindo que ele se adapte a diferentes velocidades da esteira e variações nos objetos a serem identificados. A proposta deste trabalho visa, portanto, atender às necessidades industriais de automação e flexibilidade, essenciais para a competitividade na era da Indústria 4.0.

Em resumo, este projeto propõe uma solução que se inspira nos princípios da Indústria 4.0 para melhorar o operacional e apoiar a automação nas linhas de produção. Embora não abarque todos os elementos da descentralização e interconectividade sugeridos

pela Indústria 4.0, o sistema oferece uma base sólida para rastreamento e identificação de objetos em movimento, promovendo maior controle e flexibilidade nas operações industriais. Dessa forma, ele contribui para aumentar a adaptabilidade das indústrias em um ambiente de produção.

1.2 Obejtivos Gerais

Desenvolver e implementar um sistema de visão computacional baseado no ROS (Robot Operating System) para a identificação e rastreamento de objetos em movimento em uma esteira industrial, visando otimizar a automação e aumentar a precisão e eficiência nas linhas de produção. Este sistema deve ser capaz de capturar e processar imagens em tempo real, detectar marcadores ArUco nos objetos e transmitir informações para outros componentes do sistema.

1.3 Estrutura do Documento

Este trabalho está organizado em seis capítulos, incluindo está introdução.

O **Capítulo 2** aborda a Fundamentação teórica, onde são explorados os conceitos relevantes para o contexto desse projeto, como industrias 4.0[1], o sistema operacional robótico ROS, princípios de visão computacional, e a aplicação de arUcos para a detecção de objetos. Além disso, esse capítulo discute o uso de árvores de comportamento (Behavior Trees)[2] para controle de sistemas robóticos.

No **Capítulo 3**, é detalhada a Metodologia adotada no desenvolvimento do sistema, incluindo a configuração dos dispositivos uttilizados, como a câmera Basler[3] e o manipulador UR10, além dos processos de calibração e simulação. Este capítulo também explica a integração entre os componentes e as ferramentas utilizadas para implementar o sistema.

O **Capítulo 4** trata do controle de qualidade, onde é apresentado a lógica e implementação do método de monitoramento das peças.

No Capítulo 5, são expostos os resultados e mediante esses, são feitas analises se dos resultados obtidos em relação aos objetivos.

Por fim, no **capítulo 6** são explicadas as contribuições do trablho, além de sugestões para estudos futuros.

[1] Industrias 4.0: <https://www.sap.com/brazil/products/scm/industry-4-0/what-is-industry-4-0.html>
[2] Behavior Trees: <https://www.behaviortree.dev>
[3] Basler: <https://www.baslerweb.com/en-us/>

2 Fundamentação Teorica

Neste capítulo, apresentam-se os principais conceitos e tecnologias que embasam este trabalho. Serão discutidos os fundamentos da Indústria 4.0 e sua influência na automação e integração de processos industriais. Também são explorados tópicos como visão computacional, ROS e a aplicação de marcadores fiduciais ArUco para identificação de objetos. Esses conceitos fornecem o embasamento necessário para o desenvolvimento do sistema proposto, destacando o papel da tecnologia na modernização das linhas de produção.

2.1 Indústria 4.0

A Indústria 4.0 é uma revolução de como as fábricas e sistemas de produção operam, impulsionada pela digitalização e integração de tecnologias emergentes, como Internet das Coisas (IoT), sistemas ciberfísicos, big data, visão computacional e automação robótica. Esses conceitos permitem que dispositivos conectados, como câmeras industriais e sensores, capturem e processem dados em tempo real, viabilizando a automação de processos de monitoramento contínuo e controle de qualidade. Essa transformação traz benefícios significativos, como o aumento da flexibilidade, a redução de custos operacionais e a melhoria da qualidade dos produtos, tornando as fábricas mais inteligentes e adaptáveis às mudanças no mercado (CSALODI et al., 2021).

A aplicação de tecnologias da Indústria 4.0 possibilita o desenvolvimento de plataformas inteligentes para monitoramento contínuo e controle de sistemas industriais. Essas plataformas utilizam câmeras de alta precisão e algoritmos de visão computacional para identificar e rastrear objetos em linhas de produção, otimizando o monitoramento de processos e permitindo respostas rápidas a falhas ou anomalias (ZEMLA et al., 2023). Com o uso de realidade aumentada e protocolos de comunicação avançados, esses sistemas facilitam a integração dos dados de câmeras e sensores, garantindo que as operações possam ser ajustadas dinamicamente e operem de forma autônoma, mantendo a segurança e eficiência (ZEMLA et al., 2023).

A integração de algoritmos de otimização também desempenha um papel fundamental na eficiência dos processos produtivos na Indústria 4.0. Esses algoritmos, combinados com sistemas de visão computacional, permitem que os sistemas ciberfísicos e os robôs tomem decisões em tempo real com base em dados visuais e de sensores, adaptando-se automaticamente às necessidades de produção. Com isso, a produtividade aumenta e o consumo de recursos diminui, contribuindo para a sustentabilidade do processo industrial (CSALODI et al., 2021).

Além disso, a Indústria 4.0 evoluiu para a Indústria X.0, que incorpora tecnologias mais avançadas, como gêmeos digitais e computação de borda. A aplicação de gêmeos digitais permite que sistemas visuais e de monitoramento sejam simulados e otimizados antes de serem implementados, enquanto a computação de borda assegura que os dados capturados por câmeras e sensores sejam processados localmente e com maior velocidade, aprimorando a resposta em tempo real (ONU, 2023).

Em fábricas inteligentes, o uso de robôs e sistemas autônomos combinados com visão computacional se torna cada vez mais comum. Sistemas ciberfísicos que integram câmeras e sensores de alta precisão facilitam o gerenciamento de linhas de produção em tempo real, permitindo a adaptação imediata às variações de demanda e às mudanças nas condições de operação (CHEN et al., 2017). Um exemplo desse conceito pode ser visto na linha de embalagem automatizada estudada por Chen et al. (2017), onde sensores e câmeras de monitoramento em tempo real, combinados com robôs, aumentam a eficiência e precisão dos processos, permitindo maior controle de qualidade e menor desperdício de material.

Dessa forma, a Indústria 4.0 redefine os processos de produção ao integrar tecnologias digitais, sensores visuais e sistemas inteligentes, proporcionando um ambiente onde robôs, câmeras e algoritmos de visão computacional colaboram para maximizar a eficiência e a sustentabilidade das operações industriais.

2.2 *Robot Operating System 2 (ROS2)*

O *Robot Operating System (ROS)*, traduzindo para o português sistema operacional de robôs, surgiu como uma plataforma para unificar o desenvolvimento de robôs autônomos, oferecendo um conjunto de bibliotecas e ferramentas que facilitam a comunicação e a coordenação entre os diferentes componentes de um sistema robótico. O ROS 1, precursor do ROS 2[1], focou na aceleração da pesquisa, disponibilizando uma infraestrutura de código aberto para desenvolvimento experimental. No entanto, seu design inicial não considerava requisitos industriais críticos, como segurança robusta, operação em redes heterogêneas e suporte em larga escala para sistemas de produção (MACENSKI et al., 2022).

2.2.1 A Transição do ROS 1 para o ROS 2

A transição para o ROS 2 foi impulsionada pelas limitações do ROS 1 em atender a demandas comerciais e industriais, onde sistemas autônomos operam em ambientes complexos e imprevisíveis. O ROS 2 foi projetado com base no *Data Distribution Service (DDS)*, um padrão de comunicação amplamente utilizado em sistemas críticos, o que permitiu uma arquitetura mais modular e escalável, adequada para operações em

[1] Documentação do ROS: <https://docs.ros.org/en/humble/index.html>

tempo real e com alta confiabilidade. Essa transição introduziu melhorias fundamentais, como descoberta automática entre nós, suporte a sistemas operacionais múltiplos (Linux, Windows e macOS) e configurações de segurança integradas (MACENSKI et al., 2022).

2.2.2 Conceitos e Princípios do ROS 2

O design do ROS 2 reflete princípios fundamentais de *distribuição, modularidade e abstração*. A abordagem distribuída permite que os componentes de um sistema robótico funcionem de forma independente, promovendo a resiliência e escalabilidade do sistema. O princípio da modularidade, inspirado no paradigma UNIX de "fazer uma coisa bem", fragmenta o sistema em pacotes e bibliotecas reutilizáveis, o que facilita a colaboração e o desenvolvimento paralelo. A abstração garante que os componentes possam ser substituídos ou atualizados sem interferir na integridade do sistema global (MACENSKI et al., 2022).

2.2.3 Funcionamento do Robot Operating System 2 (ROS2)

O ROS2 é uma estrutura de software para o desenvolvimento de aplicações robóticas, projetada para suportar ambientes industriais e de pesquisa. Ele é baseado em uma arquitetura modular e distribuída, onde vários componentes, chamados de *nós*, interagem entre si para realizar tarefas robóticas complexas. Abaixo, segue uma descrição detalhada dos principais conceitos e componentes do *ROS2*, e como eles trabalham juntos para fornecer uma plataforma robusta e flexível[2].

2.2.3.1 Arquitetura de Nós e Comunicação

Existem três elementos principais na comunicação entre nós:

- **nó** - um processo executável que realiza uma função específica, cada nó pode *publicar* ou *subscrever* a tópicos;
- **tópico** - um canal de comunicação para troca de mensagens que funciona como um intermediário entre nós publicadores e nós subscritores;
- **mensagem** - a unidade de dados que é transmitida entre nós, cada mensagem possui um formato específico e contém informações que os nós compartilham.

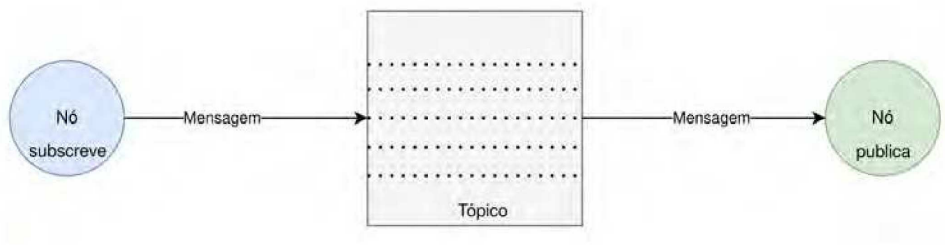
A comunicação entre nós no ROS segue um modelo de publicação e subscrição. No processo, um nó *publicador* envia mensagens para um tópico específico, enquanto um nó *subscritor* recebe essas mensagens ao subscrever-se ao mesmo tópico. Esse fluxo permite a comunicação de dados de maneira assíncrona, onde múltiplos nós podem escutar ou enviar informações sem interrupções e é descrito a seguir.

[2] Documentação do ROS: <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html>

1. **Publicação**: um nó que deseja compartilhar informações cria uma mensagem e a envia para um tópico.
2. **Subscrição**: um nó que deseja receber informações se subscreve a um tópico específico para escutar as mensagens que estão sendo publicadas.
3. **Entrega**: o tópico atua como um intermediário e encaminha a mensagem para todos os nós subscritores.

Na Figura 1 exemplifica-se o processo de comunicação entre nós no ROS. O nó à esquerda é o *publicador*, enviando uma mensagem para o tópico, que armazena e redireciona para o nó à direita, que é o *subscritor*.

Figura 1 – Comunicação dos nós



Fonte: Autoria própria

Esse modelo de comunicação oferece várias vantagens:

- **modularidade** - cada nó funciona de forma independente, facilitando a modularidade do sistema;
- **escalabilidade** - novos nós podem ser adicionados para receber ou publicar informações em tópicos sem alterar a estrutura de outros nós;
- **flexibilidade** - a comunicação assíncrona permite que os nós operem em diferentes taxas de execução.

No contexto de uma fábrica inteligente, como o projeto de Indústria 4.0 em um laboratório, esse modelo permite que diferentes dispositivos, como câmeras, braços robóticos e sensores, troquem informações em tempo real. Por exemplo, um nó associado a uma câmera pode publicar dados de imagem em um tópico, enquanto outro nó responsável pelo processamento de visão computacional recebe essas informações para realizar o controle de um braço robótico.

2.2.3.2 Modularidade e Contêineres Docker

Outro aspecto significativo do *ROS2* é o suporte a arquiteturas distribuídas e modulares. No contexto de sistemas industriais, essa modularidade permite a integração rápida de novos dispositivos e sensores sem impactar o funcionamento dos componentes existentes. Além disso, o uso de contêineres[3], como o Docker, permite que cada componente do sistema funcione de maneira isolada, facilitando a manutenção e a reconfiguração do sistema conforme as demandas do ambiente industrial. Essa abordagem é especialmente útil para sistemas de visão computacional, onde câmeras e sensores podem ser gerenciados independentemente e integrados ao sistema de forma ágil.

Com sua arquitetura modular, comunicação confiável através do *DDS*, segurança robusta e suporte para contêineres, o *ROS2* se posiciona como uma plataforma poderosa para aplicações robóticas na Indústria 4.0. Ele permite a integração de sistemas de visão computacional e dispositivos robóticos, fornecendo uma infraestrutura adaptável para monitoramento e controle de processos em tempo real, tornando-se uma solução essencial para ambientes industriais dinâmicos e de alta demanda.

2.2.4 Comunicação e Integração em Sistemas Robóticos

O ROS 2 adota padrões de comunicação por meio de *tópicos, serviços e ações* para suportar diferentes padrões de troca de informações entre os nós. Esses mecanismos de comunicação são organizados em uma arquitetura de grafo computacional, onde cada nó representa uma unidade funcional do robô. Através de configurações de qualidade de serviço (*Quality of Service* - QoS), o ROS 2 permite otimizar a transmissão de dados de acordo com as condições de rede e as necessidades de cada aplicação. Além disso, com a integração de *nós de ciclo de vida* (*lifecycle nodes*), o sistema possibilita o controle sobre o estado de cada componente, aumentando a eficiência e a segurança na execução de missões (MACENSKI et al., 2022).

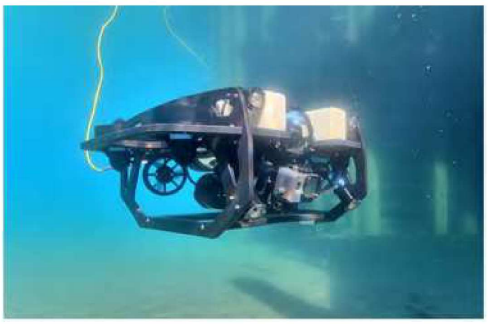
2.2.5 Aplicações Práticas do ROS 2

Estudos de caso em diferentes domínios — como robôs de quatro patas em ambientes naturais (Ghost Robotics), veículos submersíveis (Mission Robotics), drones (Auterion), e robôs espaciais (NASA VIPER) — evidenciam como o ROS 2 tem acelerado o desenvolvimento e a implementação de sistemas robóticos (ver Figura 2). Esses estudos mostram que a estrutura modular e o suporte para múltiplos dispositivos e cenários do ROS 2 permite que as empresas adaptem rapidamente seus sistemas para aplicações específicas, aumentando sua competitividade e eficiência (MACENSKI et al., 2022).

[3] Documentação do ROS: <https://www.ibm.com/br-pt/topics/containers>

Figura 2 – Estudos de caso de sistemas robóticos implementados em terra, água e ar.

(a) Robô Vision-60 da Ghost Robotics.



(b) Robô submersível da Mission Robotics.

(c) Robôs OTTO 100 da OTTO Motor.

(d) Drone da Auterion movido pela Freefly.

Fonte: Adaptado de (MACENSKI et al., 2022).

O ROS 2 representa um marco na evolução do software de robótica, sendo uma ferramenta robusta que facilita o desenvolvimento de sistemas distribuídos e seguros, aplicáveis em setores industriais e exploratórios. Sua arquitetura escalável e modular, somada à comunidade ativa de desenvolvedores, torna-o uma base essencial para futuras inovações em robótica. À medida que novos recursos e otimizações são desenvolvidos, o ROS 2 continuará impulsionando a indústria robótica, possibilitando o surgimento de soluções cada vez mais avançadas e confiáveis.

2.3 Visão Computacional

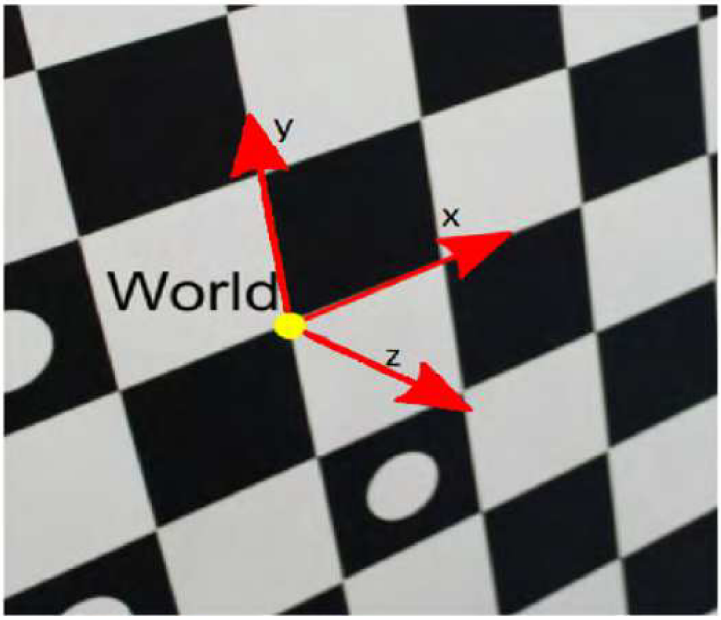
No contexto de visão computacional o objetivo é desenvolver métodos para que computadores e sistemas automatizados interpretem e compreendam imagens e vídeos, aproximando-se da capacidade de percepção visual humana. A visão computacional é fundamental em diversas aplicações, desde robótica até sistemas de controle de qualidade em indústrias, e requer uma combinação de processamento de imagem, aprendizado de máquina e modelos geométricos (SHAHRIA et al., 2022).

2.3.1 Aplicações na Robótica

Na robótica, a visão computacional tem se mostrado uma tecnologia de grande relevância para permitir que robôs interajam de forma precisa e autônoma com o ambiente. Robôs manipuladores, por exemplo, utilizam câmeras para detectar, mapear e manipular objetos, tornando-se cada vez mais comuns em tarefas de inspeção, montagem e controle de qualidade. De acordo com (SHAHRIA et al., 2022), algoritmos de redes neurais, aprendizado por reforço e métodos baseados em filtros são amplamente aplicados para análise e tomada de decisão em sistemas de visão robótica. Esses métodos têm a vantagem de permitir que robôs operem em ambientes complexos, onde a percepção visual é indispensável para a realização de atividades.

2.3.2 Modelagem e Calibração de Câmeras

Figura 3 – Modelo de tabuleiro xadrez usado em calibrações



Fonte:Modelo de câmera para lentes com distorção significativa em aplicações automotivas (LELOWICZ, 2019).

A modelagem precisa das câmeras é essencial em sistemas de visão computacional, especialmente em aplicações automotivas onde lentes com alto grau de distorção são comuns, veja Figura 3. A modelagem de distorção, como o modelo de divisão e o modelo Brown-Conrady, permite corrigir as distorções radiais e tangenciais que afetam a precisão da imagem capturada (LELOWICZ, 2019). Através de calibração com tabuleiros de xadrez, por exemplo, é possível ajustar os parâmetros da câmera para uma interpretação precisa das imagens, sendo esse processo fundamental em câmeras de visão ampla utilizadas na indústria automotiva, onde a precisão é imprescindível (LELOWICZ, 2019).

2.3.3 Classificação e Avaliação de Qualidade

Além de aplicações na robótica e automação, a visão computacional é uma ferramenta poderosa para a inspeção e controle de qualidade, como ilustrado por (SUPRIJANTO; RAKHMAWATI; YULIASTUTI, 2011) no contexto da avaliação de chá preto. A partir de características morfológicas e de cor extraídas das imagens dos grânulos de chá, um sistema de visão computacional foi desenvolvido para categorizar a qualidade em diferentes classes, utilizando redes neurais artificiais. Essa abordagem, além de substituir o processo de inspeção visual tradicional, proporciona maior precisão e eficiência, demonstrando o potencial da visão computacional para inspeções industriais (SUPRIJANTO; RAKHMAWATI; YULIASTUTI, 2011).

Apesar dos avanços, a visão computacional enfrenta desafios, como as limitações de iluminação, resolução limitada e obstruções no campo de visão. Esses fatores impactam diretamente a precisão dos sistemas de visão, tornando essencial a pesquisa de novas soluções que aumentem a robustez e a adaptabilidade das câmeras e dos algoritmos utilizados (SHAHRIA et al., 2022). A integração eficiente entre sensores e algoritmos adaptativos continua sendo um foco de pesquisa para superar essas barreiras e expandir as capacidades dos sistemas de visão computacional.

2.4 Árvores de Comportamento (*Behavior Trees*)

As Árvores de Comportamento (*Behavior Trees* - BTs) surgiram na indústria de jogos como um modelo para controlar personagens não-jogáveis, mas sua modularidade e flexibilidade rapidamente atraíram a atenção da robótica (GUGLIERMO et al., 2024). No contexto robótico, as BTs são usadas para controlar robôs que operam em ambientes dinâmicos, pois oferecem uma estrutura reativa e adaptável que permite ao robô reagir a eventos inesperados e se recuperar de erros sem interromper sua tarefa (TEAM, 2023). Além disso, as BTs facilitam o design de comportamentos complexos e reutilizáveis, essenciais para robôs em aplicações críticas, como ambientes industriais e colaborativos (HEPPNER, 2023).

2.4.1 Estrutura e Definição das Árvores de Comportamento

As BTs são compostas por diferentes tipos de nós que controlam a execução das ações. *Nós de controle*, controlam a sequência de execução dos nós filhos e incluem tipos como Sequência (executa todos os filhos em ordem) e *Fallback* (busca executar nós até encontrar um sucesso). Esses nós simplificam o fluxo de controle e permitem que a árvore se adapte a diferentes cenários com reações específicas (GUGLIERMO et al., 2024). *Nós de Execução* que são divididos entre *Nós de Ação* que executam comandos específicos e

Nós de Condição que avaliam proposições. esses nós executam as tarefas que o robô deve realizar e pelas condições que devem ser verificadas antes da execução (OGREN, 2018).

Essa organização modular permite que as BTs sejam escaláveis e flexíveis, facilitando a reutilização de subárvores para tarefas complexas. Ao contrário das Máquinas de Estado Finito (FSMs), que apresentam explosão de estados em cenários complexos, as BTs favorecem um design limpo e modular (HEPPNER, 2023; TEAM, 2023).

2.4.2 Aplicações e Coordenação em Equipes de Robôs Heterogêneos

As BTs não se limitam ao controle de robôs individuais; elas também podem ser aplicadas em equipes de robôs heterogêneos, coordenando tarefas em ambientes industriais e colaborativos. Um exemplo dessa abordagem é o conceito de BTs Distribuídas que permite a delegação de subárvores entre robôs, de acordo com suas capacidades específicas. Isso é feito por meio dos nós Shovables e Slots, que permitem o compartilhamento de tarefas entre robôs de forma adaptativa, aumentando a eficiência do sistema como um todo (HEPPNER, 2023). Essa abordagem é particularmente útil para ambientes onde múltiplos robôs devem colaborar em tempo real, como em operações de transporte ou manipulação de objetos (HEPPNER, 2023).

2.4.3 Métricas para Avaliação das BTs

Para que as BTs cumpram seu papel em contextos variados, é essencial avaliar seu desempenho com base em métricas objetivas. As métricas são divididas em funcionais e não-funcionais (GUGLIERMO et al., 2024) e são apresentadas a seguir.

2.4.3.1 Métricas Funcionais

- **Uso de Recursos**: utilizada para avaliar a eficiência de uma BT em termos de tempo de execução e consumo de energia, especialmente relevante para robôs que operam de forma contínua.
- **Taxa de Sucesso**: utilizada para medir a frequência com que a BT atinge seus objetivos, proporcionando uma visão sobre sua eficácia em cenários de alto risco.
- **Contagem de Estados Inseguros**: utilizada para medir a frequência com que o robô entra em estados perigosos, fornecendo insights sobre a segurança do comportamento.
- **Frequência de Verificação de Condição**: utilizada para medir a frequência de avaliação dos nós de condição, indicando a reatividade da BT em cenários dinâmicos (GUGLIERMO et al., 2024).

2.4.3.2 Métricas Não-Funcionais

- **Dimensões da Árvore**: incluem altura, largura e número de nós, sendo indicadores da complexidade e modularidade da árvore.
- **Granularidade de Ação**: refere-se ao nível de decomposição das ações em sub-ações menores, proporcionando flexibilidade na construção de comportamentos complexos (GUGLIERMO et al., 2024).

2.5 Propriedades Funcionais e Não-Funcionais das BTs

2.5.0.1 Propriedades Funcionais

- **Reatividade**: refere-se à capacidade da BT de responder a mudanças no ambiente de forma rápida e eficiente, característica essencial para robôs em ambientes dinâmicos (GUGLIERMO et al., 2024).
- **Corretude e Robustez**: refere-se à capacidade da BT de executar o comportamento correto mesmo diante de perturbações ambientais, garantindo que o robô continue operando de forma segura e eficaz (GUGLIERMO et al., 2024).

2.5.0.2 Propriedades Não-Funcionais

- **Modularidade**: permite que as subárvores sejam independentes e facilmente reutilizáveis em diferentes contextos, facilitando a manutenção e escalabilidade do sistema (HEPPNER, 2023).
- **Interpretabilidade**: facilita a compreensão do comportamento do robô, tornando o sistema mais acessível para operadores humanos (GUGLIERMO et al., 2024).

2.5.1 Exemplos de Aplicações e Cenários Reais

As BTs são aplicadas em uma variedade de contextos na robótica, desde operações de transporte e manipulação em fábricas até robôs colaborativos em ambientes com interação humano-robô. Em indústrias, a modularidade e eficiência são primordiais, enquanto em ambientes colaborativos, a segurança e a reatividade ganham maior relevância (HEPPNER, 2023). Para robôs autônomos em ambientes complexos, como veículos autônomos, todas as propriedades - corretude, robustez, eficiência, segurança e reatividade - são igualmente importantes para garantir uma operação segura e eficaz (GUGLIERMO et al., 2024).

A padronização de métricas para avaliação de BTs é fundamental para permitir comparações consistentes entre diferentes abordagens, promovendo avanços na aplicação das BTs em robótica.

2.6 Marcadores Fiduciais

Marcadores fiduciais são elementos visuais projetados para serem detectados e rastreados por sistemas de visão computacional, facilitando a localização e identificação de objetos na cena. Eles são amplamente utilizados em robótica e realidade aumentada, onde um sistema visual precisa de pontos de referência para estimar a posição e a orientação dos objetos no espaço tridimensional (HUANG; GRIZZLE, 2017).

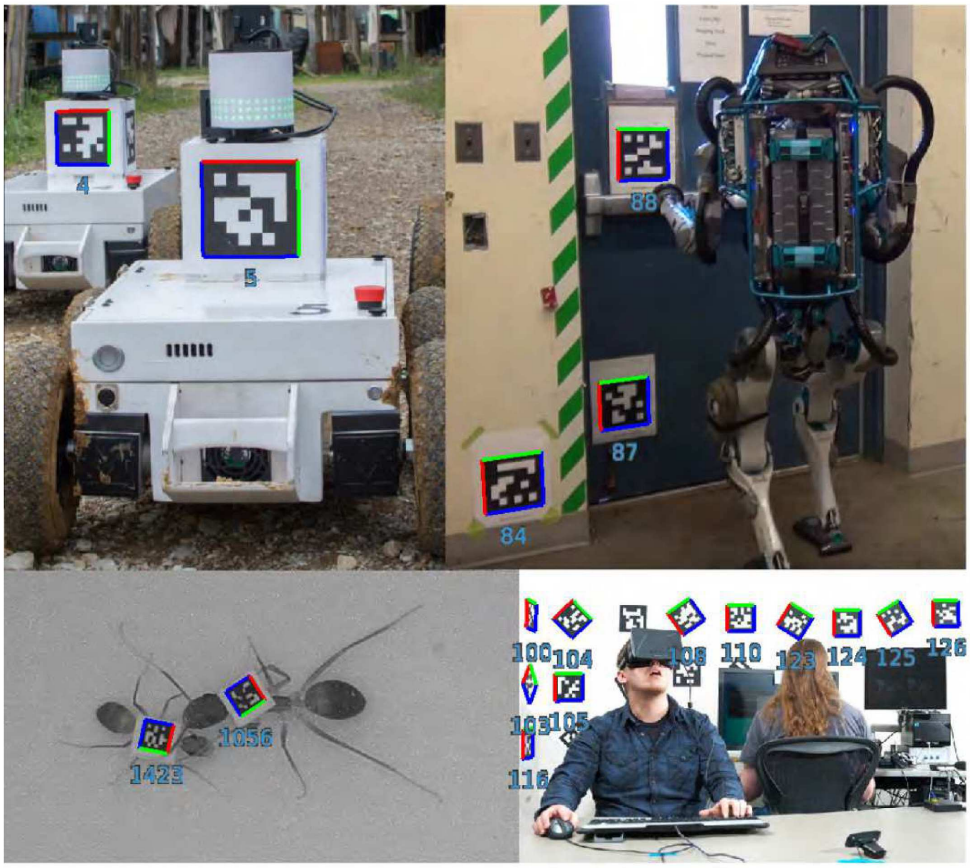
O uso de marcadores fiduciais em robótica e sistemas de calibração permite a criação de mapas tridimensionais precisos e auxilia na navegação de robôs móveis. No caso de sistemas LiDAR, os marcadores são utilizados como pontos de referência para alinhar dados capturados por múltiplos sensores, como câmeras e LiDARs, facilitando a fusão de dados e a criação de mapas semânticos tridimensionais (HUANG; GRIZZLE, 2017). Esses sistemas são cruciais em tarefas de calibração, onde pequenas imprecisões podem levar a erros significativos na fusão de dados dos sensores, prejudicando a navegação e o mapeamento.

O sistema de marcadores AprilTag, Figura 4, por exemplo, é projetado para gerar identificadores únicos e robustos, permitindo que cada marcador seja distinguido com precisão. Baseado em uma codificação binária com distância mínima de Hamming, ele oferece uma elevada taxa de acerto, mesmo sob condições desafiadoras de iluminação ou quando parcialmente oculto (WANG; OLSON, 2016). Para garantir uma detecção robusta e eficiente, o AprilTag utiliza técnicas de detecção de bordas e segmentação adaptativa, que identificam as bordas dos marcadores e permitem o ajuste preciso dos contornos, filtrando candidatos inválidos e reduzindo o tempo de processamento (WANG; OLSON, 2016).

Além disso, o sistema AprilTag 2 apresenta aprimoramentos significativos sobre o sistema original, reduzindo a taxa de falsos positivos e otimizando o tempo de detecção. Essas melhorias foram alcançadas por meio de um novo método de segmentação de bordas contínuas, que otimiza a detecção de bordas ao agrupar pixels de fronteira entre componentes conectados de forma mais eficiente (WANG; OLSON, 2016). A decodificação rápida dos marcadores é realizada utilizando uma tabela hash pré-computada, minimizando o número de comparações e permitindo uma implementação eficiente em dispositivos de baixa potência (WANG; OLSON, 2016).

Estudos demonstraram que o sistema AprilTag 2 melhora significativamente a precisão e a robustez da detecção em comparação ao sistema original, tornando-o uma opção preferida em aplicações de tempo real, como robôs móveis e dispositivos portáteis (WANG; OLSON, 2016).

Figura 4 – Marcadores fiduciais



Fonte: Aplicações de AprilTags em diferentes contextos, incluindo robótica e realidade aumentada (WANG; OLSON, 2016).

3 Metodologia

Neste Capítulo apresenta-se a metodologia utilizada para transformar o laboratório em um ambiente de Indústria 4.0, com um foco especial na detecção de peças por meio da câmera Basler. O sistema foi configurado para integrar os dispositivos e assegurar que a câmera Basler desempenhe um papel central na identificação precisa das peças que passarem pela esteira. A metodologia inclui a configuração dos dispositivos e sistemas, calibração da câmera, simulações para validar a eficácia da detecção e a integração dos componentes de modo a sincronizar o transporte, identificação, manipulação de peças. O objetivo é criar um ambiente coordenado, onde cada componente atue para otimizar o fluxo de trabalho e garantir a precisão na identificação e manipulação das peças.

3.1 Configuração do ambiente

A configuração do ambiente experimental visa replicar uma linha de produção flexível e autônoma, incorporando tecnologias e dispositivos robóticos para automação e controle de processos. O ambiente é organizado espacialmente para permitir o fluxo contínuo de peças, desde o transporte inicial até a alocação nos setores de destino, baseando-se em sensores, câmeras e robôs que operam de maneira coordenada.

3.1.1 Dispositivos utilizados

3.1.1.1 Manipuladores Robótico

Figura 5 – UR10

Fonte: Adaptado de *universal-robots*. Disponível em: <https://www.universal-robots.com/products/ur10-robot/>. Acesso em: [29/10/2024]

O modelo UR10 (ver Figura 5), fabricado pela Universal Robots, é um robô colaborativo (cobot) amplamente utilizado em ambientes industriais devido à sua flexibilidade, alcance e capacidade de carga. Esse tipo de robô foi desenvolvido para trabalhar de forma segura ao lado de humanos, proporcionando uma solução eficiente para automação de processos industriais que envolvem manipulação de objetos, montagem e inspeção de qualidade. Sua aplicação em setores industriais é vastamente explorada, uma vez que permite a execução de tarefas de forma autônoma ou em colaboração com operadores humanos.

O UR10 é um manipulador robótico de seis eixos, capaz de realizar operações de manipulação de alta precisão.O robô possui uma capacidade de carga de aproximadamente 10 kg, o que o torna adequado para tarefas que exigem uma manipulação robusta de objetos de variados tamanhos e pesos.

A facilidade de programação é outro destaque do UR10. Ele pode ser controlado e programado via uma interface gráfica intuitiva ou por meio de linguagens de programação compatíveis, incluindo o sistema ROS (Robot Operating System). A integração com o ROS não apenas facilita o controle sobre o UR10, mas também permite que ele seja coordenado com outros dispositivos e sensores no ambiente, como o Kinect e a câmera Basler, formando um sistema de automação coeso. Essa característica torna o UR10 versátil, permitindo que ele se adapte a diferentes tipos de tarefas e facilite a sua programação até para operadores com conhecimento técnico limitado.

No quesito segurança, o UR10 é projetado para trabalhar ao lado de humanos, contando com recursos de segurança integrados, como a detecção de colisões. Isso o torna uma boa escolha para ambientes onde a colaboração entre robôs e operadores humanos é necessária, permitindo que as operações ocorram de forma segura e minimizando os riscos de acidentes. As funções de segurança são especialmente relevantes em indústrias que buscam introduzir automação colaborativa sem comprometer o bem-estar dos trabalhadores.

3.1.1.2 Esteira

A esteira transportadora, Figura 6, sendo um dispositivo legado sem integração direta com o ROS, é controlada de forma totalmente independente do sistema central. Sua função principal é transportar as peças de um ponto a outro dentro do ambiente de trabalho, passando pela área de identificação e classificação. Contudo, devido à sua natureza de dispositivo legado, a esteira opera de forma autônoma e ininterrupta, o que exige que os demais componentes do sistema (como o UR10, o Kinect e a câmera Basler) se adaptem ao seu fluxo contínuo e não configurável.

Para acomodar essa limitação, o sistema é projetado para monitorar visualmente o transporte das peças pela esteira, ajustando o tempo e a sequência das operações de manipulação e classificação. A câmera Basler, posicionada estrategicamente acima da

Figura 6 – Esteira Industrial

Fonte: Autoria própria

esteira, realiza a captura das imagens e a classificação das peças de acordo com o setor de destino, enquanto o Kinect e o UR10 trabalham de forma sincronizada para que as peças sejam retiradas da esteira no momento adequado e alocadas em seus respectivos setores.

O uso de um dispositivo legado como a esteira exemplifica a realidade de muitas indústrias que reaproveitam equipamentos existentes para integrar-se ao conceito de Indústria 4.0. Apesar da falta de comunicação direta com o ROS, a configuração do sistema foi projetada para maximizar a utilidade da esteira, demonstrando que é possível alcançar uma operação coordenada mesmo em ambientes com dispositivos heterogêneos e sem integração direta. Essa abordagem reflete uma prática comum na indústria, onde a reutilização de dispositivos legados é frequentemente mais viável do que a substituição completa dos equipamentos.

3.1.1.3 Câmera Basler a2A1920-51gcPRO

A câmera Basler a2A1920-51gcPRO, Figura 7, da linha ace 2 R, é uma opção robusta para monitoramento visual em ambientes industriais, especialmente em aplicações de visão computacional para identificação e classificação de peças em movimento sobre uma esteira. Equipada com o sensor Sony IMX392LQR-C (CMOS de varredura progressiva

e obturador global), a câmera captura imagens sem distorção, com a calibração adequada, o que permite seu bom desempenho em ambientes dinâmicos.

Sua resolução de 1920 x 1200 pixels (2.3 MP) e taxa de até 51 FPS permitem captar detalhes das peças em movimento com precisão. A câmera utiliza uma interface Gigabit Ethernet para transmissão rápida de dados, reduzindo a latência entre captura e processamento. Ela suporta múltiplos modos de sincronização (hardware e software triggers), oferecendo flexibilidade para diferentes configurações de captura. Além disso, o software pylon de gerenciamento da câmera permite ajustes avançados de exposição, balanço de branco e ganho, otimizando a qualidade das imagens em diferentes condições de iluminação e facilitando a integração com o sistema de visão computacional.

Figura 7 – Câmera Basler a2A1920-51gcPro

Fonte: Adaptado de Basler. Disponível em: <https://docs.baslerweb.com/a2a1920-51gcpro#installation>. Acesso em: [29/10/2024]

A alimentação é compatível com Power over Ethernet (PoE), facilitando a instalação sem fonte dedicada, ou via conector de I/O com entrada de 12–24 VDC. A estrutura compacta (48.9 mm x 29 mm x 29 mm, sem montagem) e leve (menos de 105 g) possibilita instalação em braços robóticos ou suportes diversos.

No contexto do ROS2, a câmera integra-se facilmente com outros dispositivos do sistema, como o UR10 e o Kinect, utilizando o pacote Pylon ROS2. Isso permite que os dados de imagem sejam compartilhados em tópicos ROS, viabilizando a sincronização com outros elementos do sistema para manipulação automatizada e monitoramento contínuo das peças.

Tabela 1 – Características da Câmera Basler a2A1920-51gcPRO

Especificação	**Descrição**	**Detalhes**
Resolução Padrão	Imagem de 2.3 MP	1920 x 1200 pixels
Sensor	Progressive scan CMOS	Obturador global
Tamanho do Pixel	3.45 x 3.45 µm	Alta definição de imagem

Taxa de Quadros (FPS)	Rastreamento em tempo real	Até 51 FPS
Interface de Dados	Gigabit Ethernet, Fast Ethernet	1000 Mbit/s e 100 Mbit/s
Consumo de Energia	PoE e via I/O	4.8 W (PoE), 3.4 W (12–24 VDC)
Dimensões	Tamanho compacto	48.9 x 29 x 29 mm (sem montagem)
Peso	Leve	Menos de 105 g

3.1.1.4 Câmera Kinect 360

Kinect, Figura 8, anteriormente chamado de "Project Natal", é um sensor de movimentos desenvolvido para o Xbox 360 e Xbox One, em parceria com a empresa PrimeSense. A tecnologia incorpora câmeras RGB, projetores infravermelhos e detectores que mapeiam a profundidade através de cálculos estruturados de luz ou por tempo de voo. O sensor também possui um conjunto de microfones que se unem com o software e a inteligência artificial da Microsoft para permitir que o dispositivo realize reconhecimento de gestos em tempo real, reconhecimento de voz e detecção esquelética corporal de até quatro pessoas. Isso permite que o Kinect seja usado como um dispositivo de interface natural de usuário sem mãos para interagir com um sistema de computador.[1]

Figura 8 – Sensor *Microsoft* Kinect 360.

Disponível em: <https://pt.wikipedia.org/wiki/Kinect>. Acesso em: [29/10/2024]

O sensor Kinect desempenha um papel na parte de visualização e mapeamento da área de setorização. Ele identifica a pose de cada setor por meio de marcadores fiduciais ArUco, o que permite ao sistema monitorar a localização dos setores e enviar informações de posicionamento ao UR10. Assim, o UR10 é capaz de direcionar a peça ao setor correto com precisão. O Kinect também integra informações de localização de outros sensores e câmeras, assegurando que a setorização seja feita de forma coordenada e eficaz.

[1] Mais informações sobre o Kinect podem ser encontradas em: Wikipedia - Kinect.

Tabela 2 – Especificações técnicas do *Microsoft* kinect

Características	**Especificações**
Resolução do sensor RGB	$640 \times 480\ px$
Faixa de alcance do sensor de profundidade	0.8 à $4.0\ m$
Campo de visão horizontal	57
Campo de visão vertical	43
Frequência de captura	30 FPS
Precisão de medição de profundidade	± 1 cm

Fonte: Adaptado de (EL-IAITHY; HUANG; YEH, 2012)

3.1.1.5 TurtleBot2i

O TurtleBot, Figura 9, é um kit de robô pessoal de baixo custo com software de código aberto. Com o TurtleBot, você poderá construir um robô que possa dirigir pela sua casa, ver em 3D e ter potência suficiente para criar aplicativos interessantes.[2]

Figura 9 – *TurtleBot2i*

Disponível em: <https://www.turtlebot.com/turtlebot2/>. Acesso em: [29/10/2024]

O TurtleBot2i é responsável por realizar o transporte inicial dos objetos até a área de trabalho do UR10, garantindo que as peças cheguem ao ponto de manipulação de forma autônoma e segura. O TurtleBot utiliza sensores e algoritmos de planejamento de trajetória para se deslocar, facilitando a entrega dos objetos na área de trabalho do Kinect, onde inicia-se o processo de identificação.

[2] Mais informações sobre o Turtlebot podem ser encontradas em: turtlebot.

3.2 Simulação

A simulação é uma ferramenta importante para avaliar o desempenho e a integração de sistemas de visão computacional e robótica antes de sua implementação no ambiente real. Ela permite a análise de dispositivos e algoritmos em condições controladas e semelhantes ao mundo físico, identificando e corrigindo problemas sem os riscos e custos associados a falhas em sistemas físicos. Neste projeto, a simulação visa criar um ambiente virtual do laboratório *Smart Factoring*, incluindo o layout e os dispositivos presentes. Para isso, utilizamos o simulador *Gazebo*, integrado ao *Robot Operating System (ROS)*, que facilita a comunicação entre dispositivos simulados e sistemas de controle, simplificando a transição para o ambiente físico. Complementarmente, empregamos o *RViz* para visualizar em tempo real os dados dos sensores e o estado dos robôs.

3.2.1 Estrutura dos pacotes de simulação

Para desenvolver e controlar a simulação no Gazebo, foram criados dois pacotes principais em Python: `smartfactory_description` e `smartfactory_simulation`. Cada pacote possui uma estrutura de diretórios organizada para armazenar os arquivos e configurações necessários, facilitando o desenvolvimento modular e a organização dos recursos. A estrutura desses diretórios pode ser vista na Figura 10.

Figura 10 – Estrutura de diretórios dos pacotes

```
smartfactory_ws/
|-- src/
    |-- smartfactory_description/
    |   |-- meshes/
    |   |-- rviz/
    |   |-- urdf/
    |
    |-- smartfactory_simulation/
        |-- launch/
        |-- models/
        |-- world/
```

Fonte: Autoria própria

3.2.2 Configuração da Cena e Integração com o *ROS*

A fim de garantir a precisão na reprodução do laboratório, foram coletadas medidas do espaço físico, possibilitando uma criação que se assemelha à estrutura da sala. Utilizamos o repositório *Gazebo Models and Worlds Collection*[3] para compor o ambiente com objetos

[3] Disponível em: <https://github.com/leonhartyao/gazebo_models_worlds_collection>

prontos, e o arquivo `lab_smart_factory.world` foi configurado para refletir a disposição dos móveis e equipamentos de acordo com a configuração real.

A câmera Basler foi posicionada sobre a esteira, replicando o cenário industrial, permitindo testes de captura e processamento de imagens para a classificação de objetos em movimento. Para isso, foram utilizados arquivos `basler.urdf.xacro` que define o modelo STL da câmera, sua posição na cena e a configuração do plugin `libgazebo_ros_camera` que simula suas funcionalidades no *ROS*.

3.2.3 Execução da Simulação e Visualização em Tempo Real

A simulação completa é inicializada por meio de arquivos de lançamento que organizam todos os dispositivos e sua integração com o *ROS*. Estes arquivos incluem:

1. **Gazebo**: inicia o ambiente simulando o layout completo do laboratório.
2. **TurtleBot2i**: configura o robô móvel conforme sua posição e orientações.
3. **UR10**: inicializa o manipulador, assegurando que seus movimentos simulados correspondam aos comportamentos esperados na prática.
4. **Câmeras Kinect e Basler**: configura as câmeras para simular captura de imagem e profundidade.
5. **RViz**: lança o RViz, permitindo a visualização em tempo real da simulação.
6. **Transformações estáticas**: configura a sincronização dos sistemas de coordenadas para garantir uma simulação integrada.

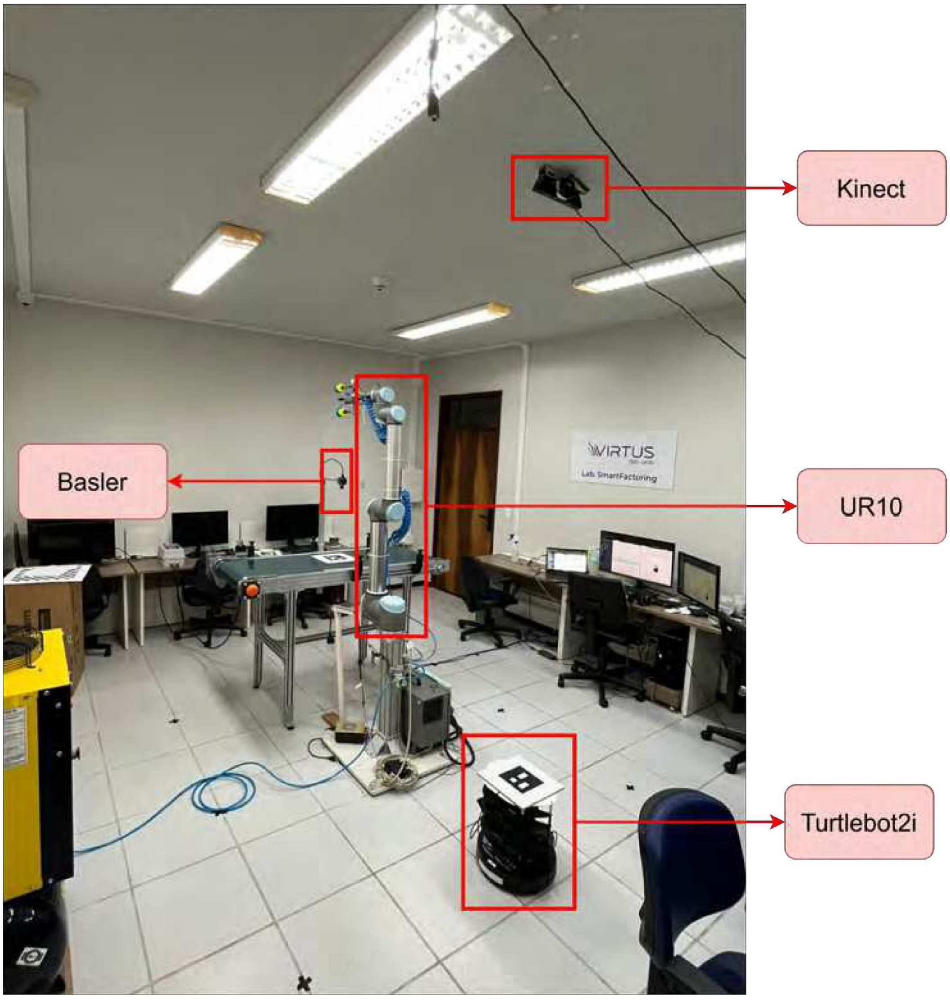
Durante a simulação, o *RViz* foi utilizado para visualizar dados dos sensores e o estado dos robôs, monitorando em tempo real pontos de calibração e posição das peças. Com isso, foi possível ajustar parâmetros da câmera Basler para capturar imagens e sincronizar a detecção e manipulação de objetos na esteira.

Nas Figuras 11, 12 e 13 são apresentadas imagens do laboratório físico *Smart Factoring*, a configuração da simulação no Gazebo e a visualização no *RViz*, demonstrando a semelhança entre a simulação e o ambiente real. Pelo RViz, é possível observar tanto o ambiente simulado quanto o ambiente real. No caso da câmera Basler, os nós de captura de imagem foram inicialmente observados pelo RViz.

3.3 Integração dos Componentes

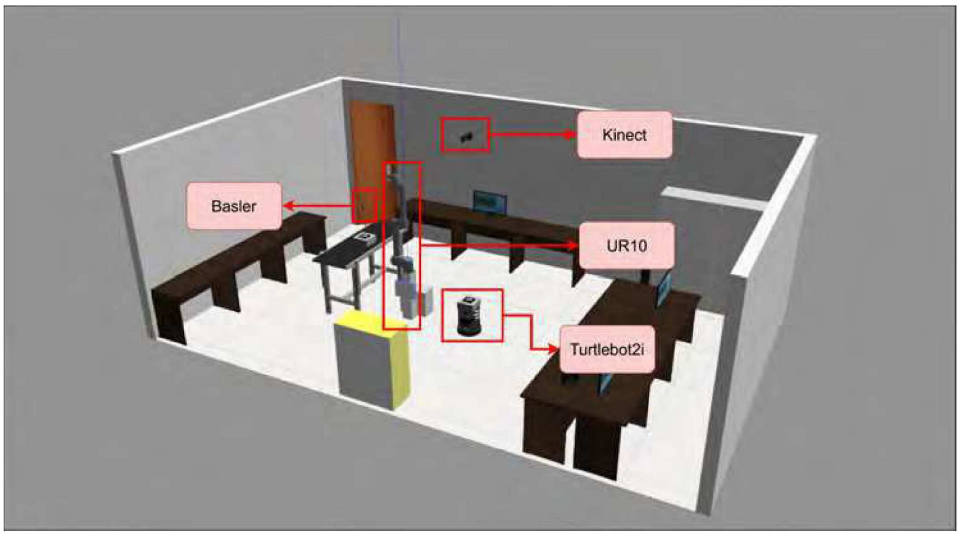
Nesta seção de Integração dos Componentes, detalha-se como os diferentes dispositivos e sensores interagem para atingir os objetivos do sistema de visão computacional

Figura 11 – Laboratório físico *Smart Factoring*.



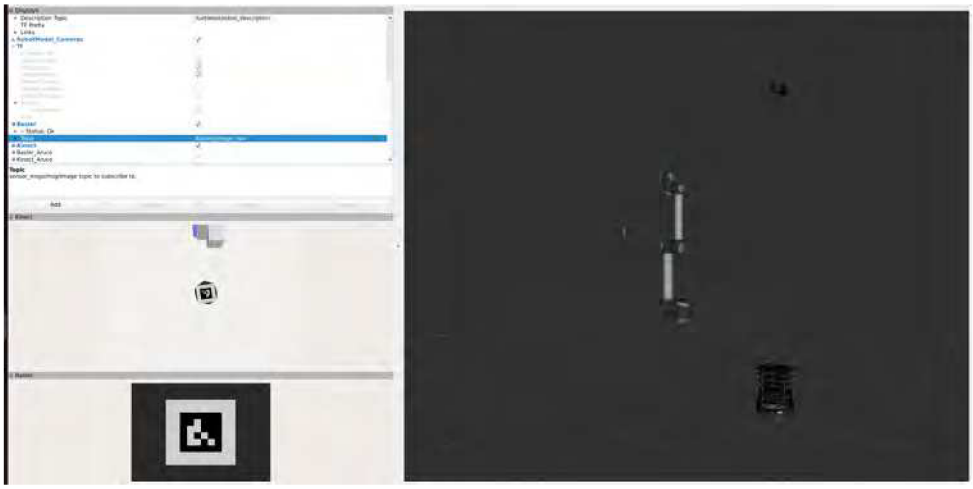
Fonte: autoria própria

Figura 12 – Imagem da Simulação do laboratório no Gazebo.



Fonte: autoria própria

Figura 13 – Imagem da Visualização no *RViz*.



Fonte: autoria própria

e automação. O foco está no fluxo de dados, na comunicação e nos comandos entre os dispositivos, destacando o papel específico de cada componente e como colaboram para a execução das tarefas. No ambiente simulado, o marcador ArUco é monitorado pela câmera posicionada acima da esteira, o que permitiu compreender o comportamento dos dispositivos antes da implementação no ambiente real. Essa simulação inicial mostrou-se uma abordagem eficaz para garantir a integridade e o funcionamento adequado dos dispositivos na aplicação prática.

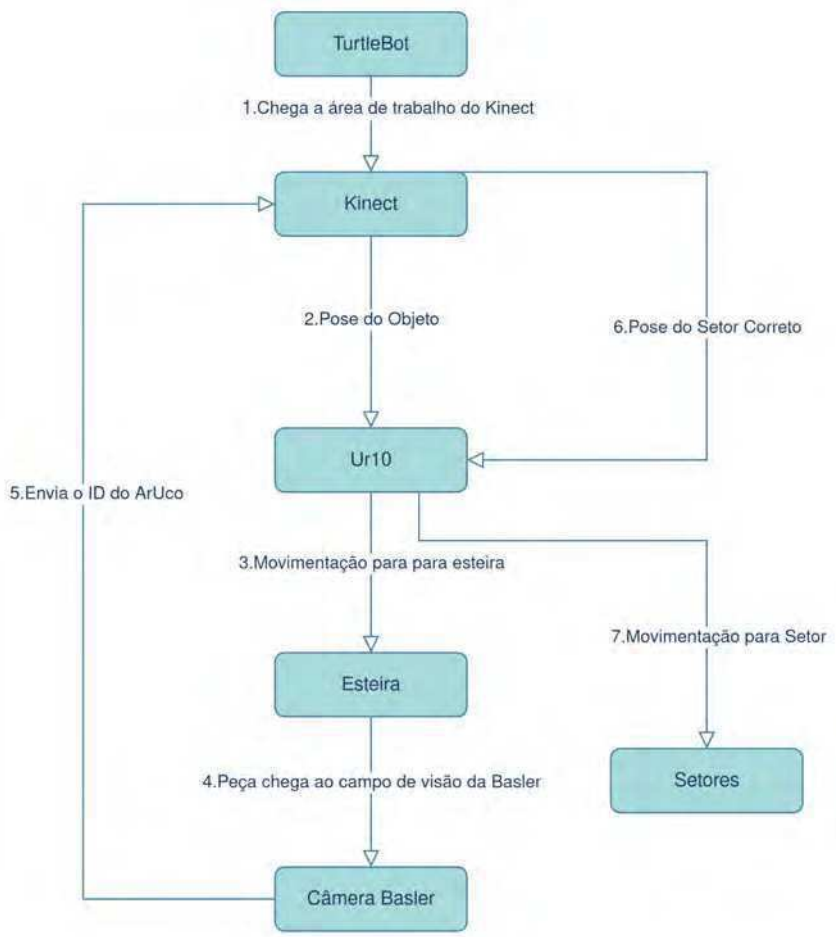
3.3.1 Arquitetura Geral do Sistema

Na Figura 14 é apresentada a arquitetura geral do sistema, incuindo os componentes físicos, bem como e a sequencia ações executadas no ambiente de trabalho. A disposição do TurtleBot, Kinect, câmera Basler, manipulador UR10 e esteira transportadora é descrita para ilustrar o fluxo de informações e dados desde a identificação inicial dos objetos até a alocação final nos setores de destino.

3.3.2 Integração do Dispositivo Legado: A Esteira Transportadora

A integração com a esteira transportadora, que opera independentemente do ROS, representa uma abordagem estratégica para a reutilização de dispositivos legados, destacando uma prática comum na indústria moderna. A escolha de manter a esteira como dispositivo legado é fundamentada pela viabilidade de reaproveitar equipamentos existentes, evitando custos adicionais de substituição e minimizando o impacto ambiental associado ao descarte de máquinas operacionais. Contudo, essa escolha apresenta desafios notáveis, especialmente pela falta de uma interface nativa para comunicação direta com o sistema central.

Figura 14 – Arquitetura geral do sistema



Fonte: Autoria própria

Esses desafios foram abordados no projeto por meio da configuração de um monitoramento visual contínuo da esteira, executado pela câmera Basler. Posicionada estrategicamente, a câmera captura imagens em tempo real das peças que se deslocam na esteira e identifica suas características para determinar o setor de destino. Todo o monitoramento é feito levando em consideração que a esteira é incapaz de se comunicar com os outros dispositivos.

Para garantir um fluxo contínuo e coordenado entre os dispositivos, foram implementadas soluções específicas para ajustar a sequência de operações ao ritmo fixo e independente da esteira. O uso de marcadores fiduciais ArUco facilita a identificação e localização das peças transportadas, permitindo que o sistema, operando no ROS, antecipe a chegada de cada peça ao ponto de manipulação. Assim, o sistema de visão computacional pode agir em sincronia com o movimento autônomo da esteira, mantendo a eficiência e precisão do processo sem interrupções.

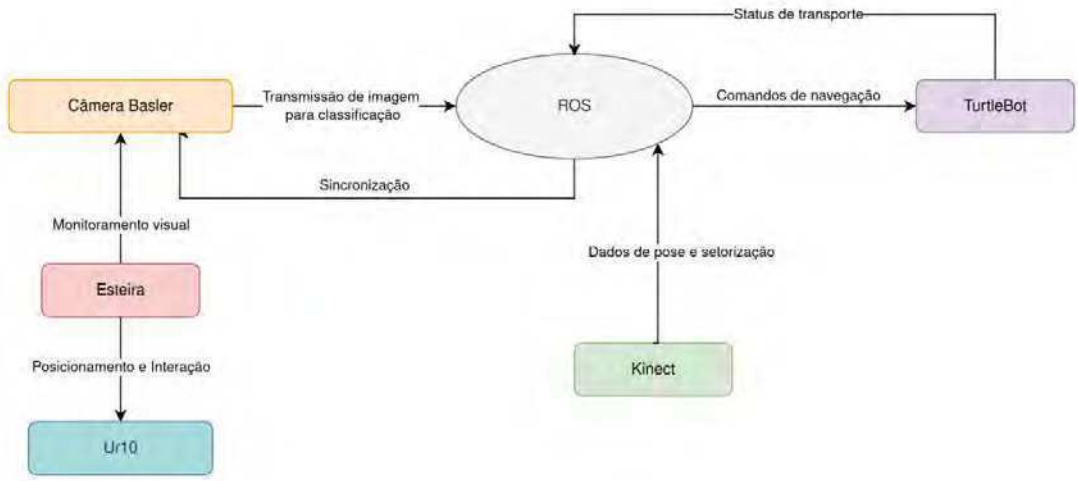
Esse tipo de integração modular e adaptativa, utilizando equipamentos legados,

reflete um paradigma chave para a Indústria 4.0, onde a modernização dos processos não implica necessariamente a substituição de todos os equipamentos existentes. Em vez disso, a adaptação dos dispositivos legados e a interoperabilidade com tecnologias avançadas, como sensores e robôs controlados por ROS, são exploradas para maximizar a eficiência e reduzir custos.

3.3.3 Comunicação entre Dispositivos usando ROS

A câmera Basler, integrada ao ROS, transmite dados de imagem por meio de tópicos específicos, que permitem o processamento e análise em tempo real. O tópico principal para envio de imagens utiliza o tipo de mensagem `sensor_msgs/Image`, que encapsula as imagens capturadas pela câmera em uma estrutura compatível com o sistema de processamento do ROS., Figura 15 Além disso, o ROS emprega o tipo de mensagem `geometry_msgs/Pose` para especificar a posição e orientação dos objetos detectados na cena, facilitando a coordenação entre a câmera e outros dispositivos. Esses tipos de mensagens são usadas para garantir que os dados visuais e posicionais da câmera sejam processados com precisão, contribuindo para o funcionamento eficiente do sistema.

Figura 15 – Comunicação dos dispositivos com o ROS



Fonte: Autoria própria

3.3.4 Fluxo de Operação e Coordenação entre Componentes

No cenário de **Setorização com mais de 2 Setores**, o TurtleBot transporta objetos para a área de trabalho do Kinect, onde cada objeto é identificado. Após a identificação, o UR10 recolhe o objeto e o posiciona na esteira. A câmera Basler identifica o objeto na esteira e determina seu setor de destino. O Kinect então registra a localização de cada setor com identificadores ArUco e comunica ao UR10, que realoca o objeto para

o setor correto. O fluxo de informações entre os dispositivos garante que a peça seja direcionada corretamente, validando a coordenação entre o UR10, câmera Basler e Kinect.

3.4 Configuração da Câmera Basler no ROS

A configuração da câmera Basler no ROS foi feita usando o `pylon_ros2_camera`. Para assegurar uma captura de imagens de qualidade, a câmera foi configurada com os seguintes parâmetros:

A resolução da câmera foi ajustada para capturar imagens a 1920x1200 pixels, com taxa de 51 fps, permitindo o monitoramento em tempo real necessário para a identificação e manipulação precisa das peças.

A câmera opera com exposição automática e controle de ganho adaptativo, ajustando-se a diferentes condições de iluminação do ambiente. Esses ajustes minimizam a necessidade de ajustes manuais, garantindo imagens consistentes. Os parâmetros de calibração da câmera foram salvos em um arquivo YAML, especificado no campo `camera_info_url`, permitindo que o ROS2 carregue automaticamente as configurações de calibração a cada inicialização, corrigindo distorções e assegurando a precisão da captura.

3.4.1 Detecção de ArUcos e Extração de Pose

A detecção de marcadores ArUco e a extração de pose foram realizadas utilizando o pacote `aruco_ros`, que processa as imagens da câmera Basler em tempo real (ver figura 16).

Figura 16 – Deteção de Aruco usando a câmera basler.

Fonte: Autoria própria

Primeiramente, o algoritmo localiza os marcadores ArUco nas imagens, utilizando padrões visuais específicos que facilitam sua detecção, mesmo em ambientes com variabilidade de iluminação. A posição dos marcadores é calculada em relação à câmera, fornecendo dados em coordenadas tridimensionais. Para cada marcador detectado, o pacote calcula a pose — ou seja, a posição e orientação do marcador em relação à câmera —, determinando a localização do objeto na esteira. As poses dos marcadores são publicadas em tópicos ROS, permitindo que outros componentes do sistema acessem essas informações, como o controlador do UR10, que pode se inscrever no tópico em que a câmera está publicando a pose e utilizar as informações conforme necessário, dependendo da aplicação.

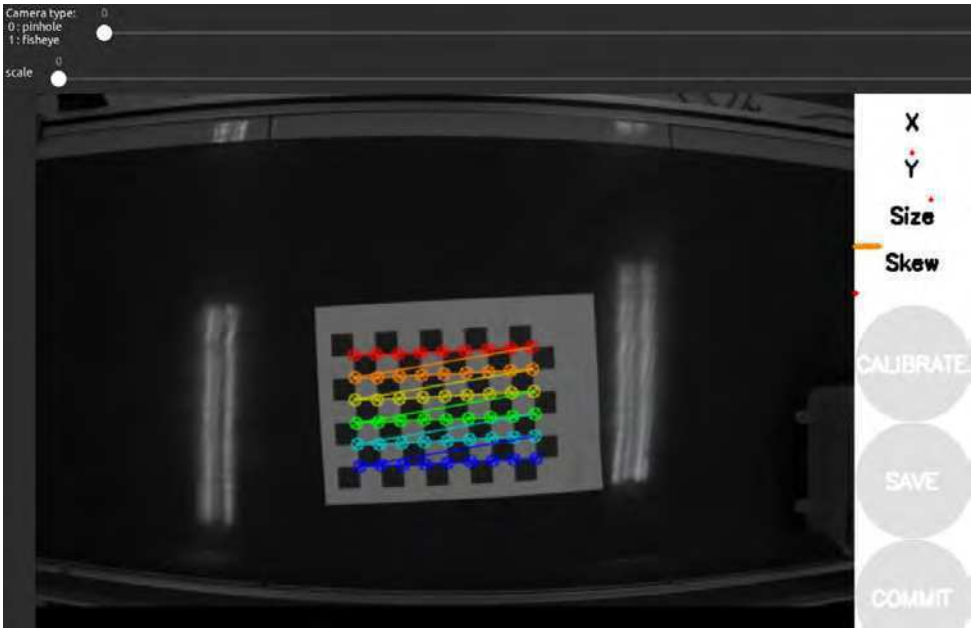
3.5 Calibração da Câmera Basler

A calibração da câmera Basler foi realizada utilizando a ferramenta de calibração de câmeras do ROS, uma interface que ajusta com precisão os parâmetros intrínsecos da câmera. Esse processo é importante em aplicações de visão computacional em ambientes industriais, onde a precisão visual impacta diretamente o reconhecimento e a manipulação de objetos. Para que a calibração funcione corretamente, é necessário que o nó do ROS responsável pela câmera esteja ativo e o arquivo de inicialização (*launch*) correspondente da câmera Basler seja iniciado. Isso permite que o ROS acesse a câmera e processe as imagens em tempo real. A calibração visa compensar as distorções introduzidas pela lente da câmera, assegurando que as imagens capturadas correspondam de forma precisa ao ambiente físico e estejam devidamente alinhadas para análise visual subsequente.

A **interface de calibração do ROS**, Figura 17, utiliza um padrão de tabuleiro de xadrez (comumente de 9x6 quadrados, cada quadrado com uma dimensão específica, com 25 mm, dependendo do setup) como referência. Este padrão é exibido em várias posições e ângulos à frente da câmera durante o processo de calibração. O software do ROS então utiliza esses pontos de referência para calcular e ajustar os parâmetros da câmera, incluindo a matriz intrínseca, os coeficientes de distorção, a matriz de retificação e a matriz de projeção. A calibração com o tabuleiro de xadrez permite que o sistema detecte pontos específicos em cada cruzamento do padrão, usados como referências para calcular as propriedades geométricas da câmera. Esses pontos ajustam as distorções da lente e garantem a precisão da posição dos objetos identificados pela câmera.

Para documentar o processo de calibração, foi realizada uma captura da imagem da câmera antes e depois da calibração. A seguir, detalhamos os parâmetros ajustados durante o processo de calibração e como cada um contribui para a precisão da visão computacional.

Figura 17 – Interface de calibração do ROS.



Fonte: Autoria própria

3.5.1 Parâmetros Ajustados na Interface de Calibração do ROS

A interface de calibração do ROS fornece diversos parâmetros que podem ser ajustados manualmente ou automaticamente, dependendo das características da lente e do sensor da câmera. Abaixo estão as descrições detalhadas de cada um desses parâmetros.

1. **Camera Type (Tipo de Câmera)**: permite a seleção entre os modelos de projeção "Pinhole"(projeção comum em câmeras industriais) e "Fisheye"(para lentes grande angulares que requerem correção adicional de distorção). Para a câmera Basler, o tipo *Pinhole* foi utilizado, pois sua lente é projetada para capturas industriais padrão.

2. ***Scale* (Escala)**: permite ajustar o nível de zoom ou escala da imagem capturada, garantindo que o padrão de calibração preencha a área de visão necessária para uma detecção precisa dos pontos. Este parâmetro é útil quando o campo de visão da câmera precisa ser ajustado para capturar o padrão de xadrez de calibração adequadamente.

3. **X e Y**: possibilita controlar a posição da imagem em relação aos eixos X e Y da tela. Ajustes nesses parâmetros garantem que o padrão de calibração esteja centralizado na imagem, o que é importante para calibrações em que o campo de visão precisa estar alinhado com o centro da câmera.

4. ***Size* (Tamanho)**: define o tamanho do padrão de calibração na captura. Um tamanho correto facilita a detecção precisa dos pontos de calibração, que é essencial para uma calibração de alta precisão dos parâmetros intrínsecos da câmera.

5. ***Skew* (Distorção Angular)**: permite ajustar qualquer distorção angular que possa estar presente, especialmente relevante quando os pixels não estão perfeitamente alinhados com os eixos principais da câmera. Este ajuste permite que a imagem seja corrigida para evitar inclinações indesejadas.

6. **Botões de Calibração**:

   - *CALIBRATE* - inicia o processo de calibração, ajustando automaticamente os parâmetros da câmera com base nas capturas do padrão de xadrez;
   - *SAVE* - salva os parâmetros ajustados em um arquivo de configuração YAML, que é utilizado posteriormente para carregar automaticamente as configurações de calibração;
   - *COMMIT* - finaliza o processo de calibração, aplicando os parâmetros ajustados permanentemente para futuras capturas de imagem.

3.5.2 Efeitos da Calibração na Qualidade da Imagem

Antes da calibração, Figura 18, as imagens capturadas pela câmera apresentavam uma distorção significativa nas bordas, especialmente perceptível em objetos posicionados nas extremidades do campo de visão. Esse efeito de distorção, conhecido como distorção radial, é comum em câmeras com lentes grande angulares e resulta em uma curvatura nas bordas da imagem. Isso pode afetar drasticamente a precisão dos sistemas de visão computacional, pois os objetos próximos às bordas tendem a sofrer alterações de forma e tamanho, introduzindo erros nas medições de perspectiva e proporção.

Figura 18 – Imagem antes da calibração

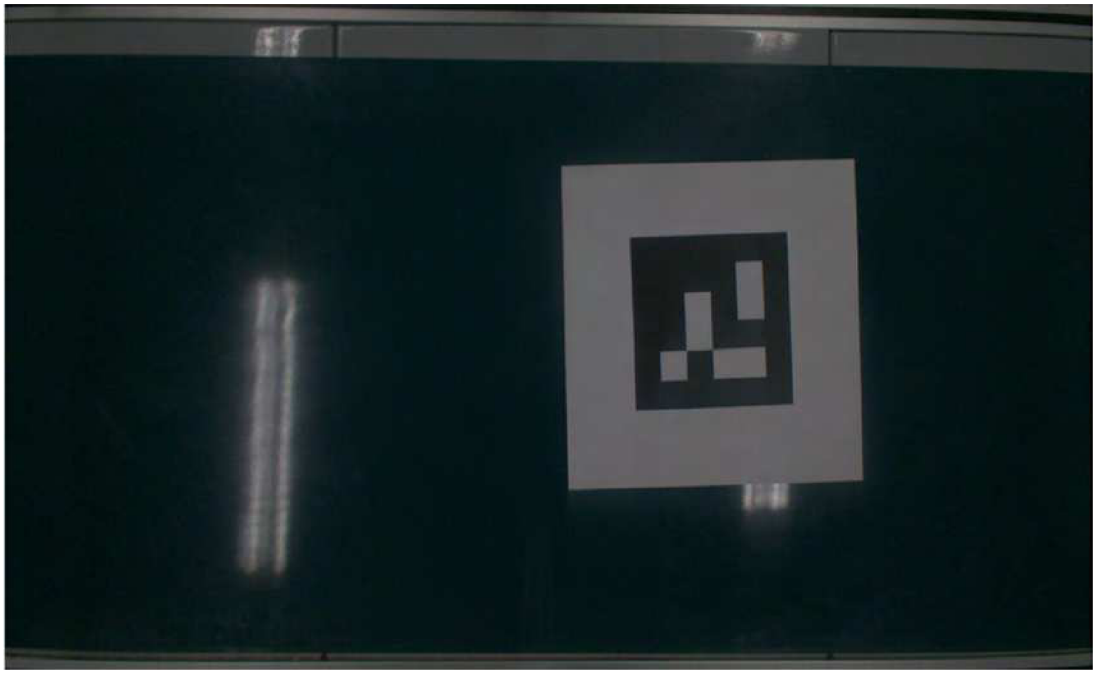
Fonte: Autoria própria

Para visualizar essa distorção, foi utilizado o software **Pylon Viewer**, desenvolvido pela Basler. O Pylon Viewer é uma ferramenta poderosa que acompanha a câmera Basler e permite a visualização em tempo real das imagens capturadas, além de fornecer uma interface para ajustar diversos parâmetros da câmera, como exposição, ganho e configurações de calibração. Na Figura 18, é possível observar a imagem inicial, antes da aplicação dos ajustes de calibração feitas usando o ROS2, onde a distorção nas bordas é evidente.

O Pylon Viewer também foi o ponto de partida, pois mediante o uso do software para conferir a imagem da câmera antes do uso da câmera pelo pacote da mesma, no ROS2. Após a visualizaçã

Após a calibração, Figura 19, observou-se uma redução significativa dessas distorções. A imagem abaixo ilustra a interface de calibração do ROS em ação, com uma captura do padrão de xadrez e os ajustes em tempo real.

Figura 19 – Imagem após a calibração



Fonte: Autoria própria

Abaixo estão os parâmetros finais obtidos após a calibração:

$$\texttt{camera\_matrix} = \begin{bmatrix} 1204.46725 & 0 & 958.96128 \\ 0 & 1205.32426 & 583.07432 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\texttt{distortion\_coefficients} = \begin{bmatrix} -0.229734 & 0.075366 & 0.000241 & 0.000729 & 0 \end{bmatrix}$$

$$\texttt{rectification\_matrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\texttt{projection\_matrix} = \begin{bmatrix} 1033.39323 & 0 & 961.66173 & 0 \\ 0 & 1135.02527 & 581.2948 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

3.5.3 Impacto da Calibração no Sistema de Visão Computacional

A calibração permite que o sistema ROS2 utilize automaticamente os parâmetros ajustados para corrigir distorções em tempo real, assegurando que as imagens processadas pela câmera estejam devidamente alinhadas com as exigências do ambiente industrial. Em aplicações práticas, essa precisão é necessária para o reconhecimento de objetos, cálculo de posições e manipulação de peças em uma linha de produção. A redução das distorções minimiza erros de leitura e garante que os dados visuais capturados estejam adequados para tomada de decisões automatizadas.

Esses parâmetros são referenciados através do `camera_info_url` no arquivo YAML de configuração, o que permite que a câmera Basler a2A1920-51gcPRO opere de forma integrada com outros dispositivos ROS, como manipuladores e esteiras, mantendo a consistência de dados em todo o sistema de visão computacional.

4 Implementação do Sistema de Detecção e Rastreamento de Objetos em Esteira Industrial

Para esta implementação, o sistema de detecção e rastreamento de objetos em uma esteira industrial foi estruturado utilizando Árvores de Comportamento no ROS. A organização em árvores de comportamento permite modularizar e sincronizar as operações de captura de imagem, detecção e contagem de marcadores ArUco, garantindo a execução coordenada do sistema.

4.1 Estrutura e Funcionamento da Árvore de Comportamento

O sistema é organizado com uma árvore de comportamento composta por nós com funções específicas, que interagem para realizar a captura de imagens e a detecção de marcadores na esteira. A árvore é implementada com a biblioteca `py_trees` e tem como base o nó raiz e duas sequências principais: **SequenciaCamera** e **SequenciaAruco**.

4.1.1 Nó Raiz (Root)

O nó raiz, chamado de **Raiz** na árvore, é um nó do tipo `Parallel` configurado com a política `SuccessOnAll`. Essa configuração permite que o sistema execute as duas sequências principais (monitoramento da câmera e detecção de ArUcos) simultaneamente e sincronize os comportamentos. A política de execução paralela facilita a operação contínua e coordenada do sistema, mantendo o monitoramento e a detecção sempre ativos.

4.2 Sequências e Nós de Comportamento

A árvore é dividida em duas sequências principais, cada uma contendo nós específicos para realizar funções de detecção e contagem de marcadores. Abaixo está a descrição detalhada de cada sequência e seus nós filhos.

4.2.1 SequenciaCamera

A **SequenciaCamera** é responsável por iniciar a câmera Basler e configurar o ROS para receber os dados visuais em tempo real. Esse nó contém um único comportamento.

4.2.1.1 IniciaBaslerROS

Esse nó, implementado pela classe `StartBaslerNode`, inicializa o processo ROS para a câmera Basler. Ele usa um comando `subprocess.Popen` para abrir um terminal e executar o comando que ativa a câmera, permitindo que o ROS comece a receber as imagens capturadas. Esse nó retorna `SUCCESS` se o processo da câmera foi iniciado corretamente e `FAILURE` se houve algum problema no início. Esse passo é essencial, pois sem ele o sistema não teria dados visuais para processar.

4.2.2 SequenciaAruco

A **SequenciaAruco** é responsável pela detecção e contagem dos marcadores ArUco nas imagens capturadas pela câmera Basler. Esta sequência contém dois comportamentos principais:

4.2.2.1 ArucoDetectado

Esse nó, representado pela classe `ArucoDetectado`, verifica se um marcador ArUco foi detectado na imagem. Ele acessa a última mensagem recebida pelo tópico de ArUcos e verifica se a detecção ocorreu dentro de um intervalo de tempo específico. Se um marcador foi detectado recentemente, o nó retorna `SUCCESS`; caso contrário, retorna `FAILURE`. Esse comportamento é fundamental para garantir que o sistema esteja detectando os marcadores corretamente.

4.2.2.2 ArucoCount

Esse nó, implementado pela classe `ArucoCount`, é responsável por contar o número de vezes que cada marcador ArUco é detectado. Ele acessa a última mensagem com os IDs e poses dos marcadores detectados, armazena esses dados e contabiliza as ocorrências para cada marcador. Essa contagem permite que o sistema monitore o fluxo de objetos na esteira e forneça dados para outros componentes, como o controlador do UR10, que pode usar essas informações para realizar ações com base na presença e posição dos objetos.

4.3 Lógica e Racionalidade da Implementação

A implementação foi planejada para garantir que cada comportamento ocorra de maneira modular e sincronizada, facilitando o controle e a coordenação entre a captura de imagem e a detecção dos marcadores ArUco. A divisão em duas sequências principais (captura de imagem e detecção de ArUco) permite que o sistema de detecção e contagem opere de maneira contínua, independentemente do processo de captura de imagem.

4.3.1 Modularidade

A estrutura modular da árvore permite que cada etapa do processo seja isolada e manipulada individualmente, o que facilita a manutenção e a adição de novas funcionalidades.

4.3.2 Sincronização e Confiabilidade

Ao utilizar a política `SuccessOnAll`, a árvore garante que os processos de captura de imagem e de detecção dos marcadores ArUco estejam ativos ao mesmo tempo, promovendo uma operação confiável e sem interrupções.

4.4 Descrição do Código

- **Classe `StartBaslerNode`**: Responsável por iniciar o processo ROS da câmera Basler, utilizando um subprocesso para executar o comando de inicialização da câmera. Retorna `SUCCESS` se o processo estiver rodando, e `FAILURE` caso contrário.
- **Classe `BaslerTreeNode`**: Esta é a classe principal que gerencia as assinaturas ROS para a câmera Basler e os marcadores ArUco. Ela se inscreve nos tópicos `/basler/camera_info` e `/basler/aruco/markers` para receber informações da câmera e dos marcadores, armazenando os dados em variáveis para serem acessados pelos nós de comportamento.
- **Classe `ArucoDetectado`**: Verifica se um marcador ArUco foi detectado recentemente. Ele compara o tempo da última detecção com o tempo atual para determinar se um marcador está presente na imagem em tempo real.
- **Classe `ArucoCount`**: Conta o número de vezes que cada marcador ArUco é detectado e armazena essas informações em um dicionário, permitindo o monitoramento contínuo e a coleta de dados sobre os objetos na esteira.
- **Função `create_root`**: Cria a estrutura da árvore de comportamento, adicionando a sequência de inicialização da câmera e a sequência de detecção e contagem de ArUco. Retorna o nó raiz configurado com a política `SuccessOnAll`.
- **Função `main`**: Inicializa o nó ROS e a árvore de comportamento, configura o executor e inicia a execução do sistema. Ela também gerencia o encerramento do sistema em caso de falha ou interrupção.

Essa implementação com Árvores de Comportamento no ROS organiza a captura e processamento de imagens em uma estrutura simples. A separação das funções de

inicialização da câmera, detecção e contagem de marcadores permite que o sistema seja facilmente escalado e adaptado a novas funcionalidades.

5 Resultados

Neste capítulo, são apresentados os resultados dos testes realizados para avaliar o desempenho do sistema de detecção e rastreamento de objetos em uma esteira industrial, utilizando Árvores de Comportamento. Os testes incluem verificações iniciais do funcionamento da árvore, análise do comportamento dos nós em diferentes estados de detecção, e a coleta de dados para diferentes velocidades da esteira.

5.1 Teste Inicial: Funcionamento Completo da Árvore

No primeiro teste, foi verificado o funcionamento completo da árvore de comportamento. Em um cenário ideal, onde todos os nós executaram corretamente suas funções, a árvore retornou sucesso em todas as sequências, o que foi representado visualmente pela árvore "toda verde". Esta imagem demonstra que todos os nós nas sequências de captura da câmera e detecção de ArUco retornaram sucesso(ver Figura 20).

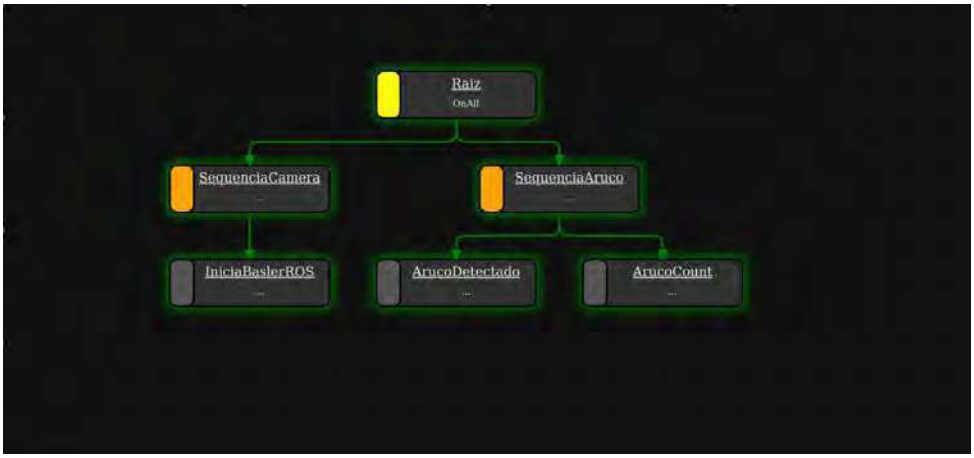


Figura 20 – Árvore de comportamento com todos os nós retornando sucesso.

5.2 Árvore Parcialmente Funcional

Outro teste realizado envolveu a execução da sequência de captura da câmera funcionando corretamente, enquanto a sequência de detecção de ArUco retornou falha, indicando que nenhum marcador ArUco foi detectado. Este cenário permite avaliar o comportamento da árvore quando apenas uma parte do sistema está operando corretamente, o que é essencial para a detecção de possíveis falhas(ver Figura 21).

Figura 21 – Árvore de comportamento com a sequência de captura da câmera funcionando e a sequência de detecção de ArUco retornando falha.

5.3 Teste com Diferentes Velocidades da Esteira

Para avaliar o impacto da velocidade da esteira na detecção dos marcadores ArUco, foram realizadas medições em três configurações de velocidade: 100% (velocidade máxima), 75% e 50%. Em cada um desses cenários, foram utilizados 7 marcadores ArUco com IDs diferentes.

Em todos os testes, todos os ArUcos foram detectados, porém, observou-se que, quanto maior a velocidade da esteira, menor foi a quantidade de poses de cada ArUco registradas pelo sistema. Isso indica que a taxa de detecção diminui conforme a velocidade da esteira aumenta, mas, ainda assim, o sistema foi capaz de detectar todos os ArUcos em todas as configurações.

5.4 Quantidade de Ticks por Velocidade

Outro dado relevante coletado foi a quantidade de ticks[1] registrados para cada uma das velocidades da esteira. Ao final da passagem de todos os ArUcos pela esteira, foi realizada coletada a quantidade de ticks para cada uma das velocidades, demonstrando o comportamento da árvore de comportamento durante todo o processo, como apresentado nas Figuras 22, 23 e 24.

5.5 Tempo de Detecção e Publicação dos ArUcos

Por fim, foi plotado o tempo de detecção de cada ArUco desde o momento em que ele foi detectado até sua publicação. Esse dado é essencial para avaliar a latência do sistema e entender o tempo necessário para processar e publicar as informações de cada

[1] Disponível em: <https://www.behaviortree.dev/docs/learn-the-basics/BT_basics>

Tabela 3 – Detecção de ArUcos em diferentes velocidades da esteira

Velocidade	**ID do ArUco**	**Número de Poses Detectadas**	**Frequência (Hz)**
50%	ID: 30	9	30
	ID: 10	10	30
	ID: 35	9	30
	ID: 25	9	30
	ID: 5	9	30
	ID: 40	10	30
	ID: 20	8	30
75%	ID: 30	5	45
	ID: 10	7	45
	ID: 35	7	45
	ID: 25	8	45
	ID: 5	8	45
	ID: 40	8	45
	ID: 20	7	45
100%	ID: 30	4	60
	ID: 10	4	60
	ID: 35	5	60
	ID: 25	5	60
	ID: 5	5	60
	ID: 40	5	60
	ID: 20	4	60

marcador ArUco. Esta análise é especialmente útil para aplicações em que a resposta em tempo real é crítica (ver Figura 25).

5.6 Análise dos Resultados

Os testes realizados mostram que o sistema é funcional e confiável, mesmo em diferentes velocidades da esteira. Embora a quantidade de poses detectadas de cada ArUco diminua conforme a velocidade aumenta, o sistema foi capaz de detectar todos os ArUcos em todas as configurações. A árvore de comportamento demonstrou-se robusta e eficaz para lidar com os diferentes estados do sistema, e o tempo de detecção até a publicação dos ArUcos apresentou valores adequados para aplicações industriais.

Esses resultados indicam que o sistema proposto é adequado para aplicações em ambientes industriais, onde é necessária uma detecção contínua e precisa de objetos em movimento, como em uma esteira transportadora.

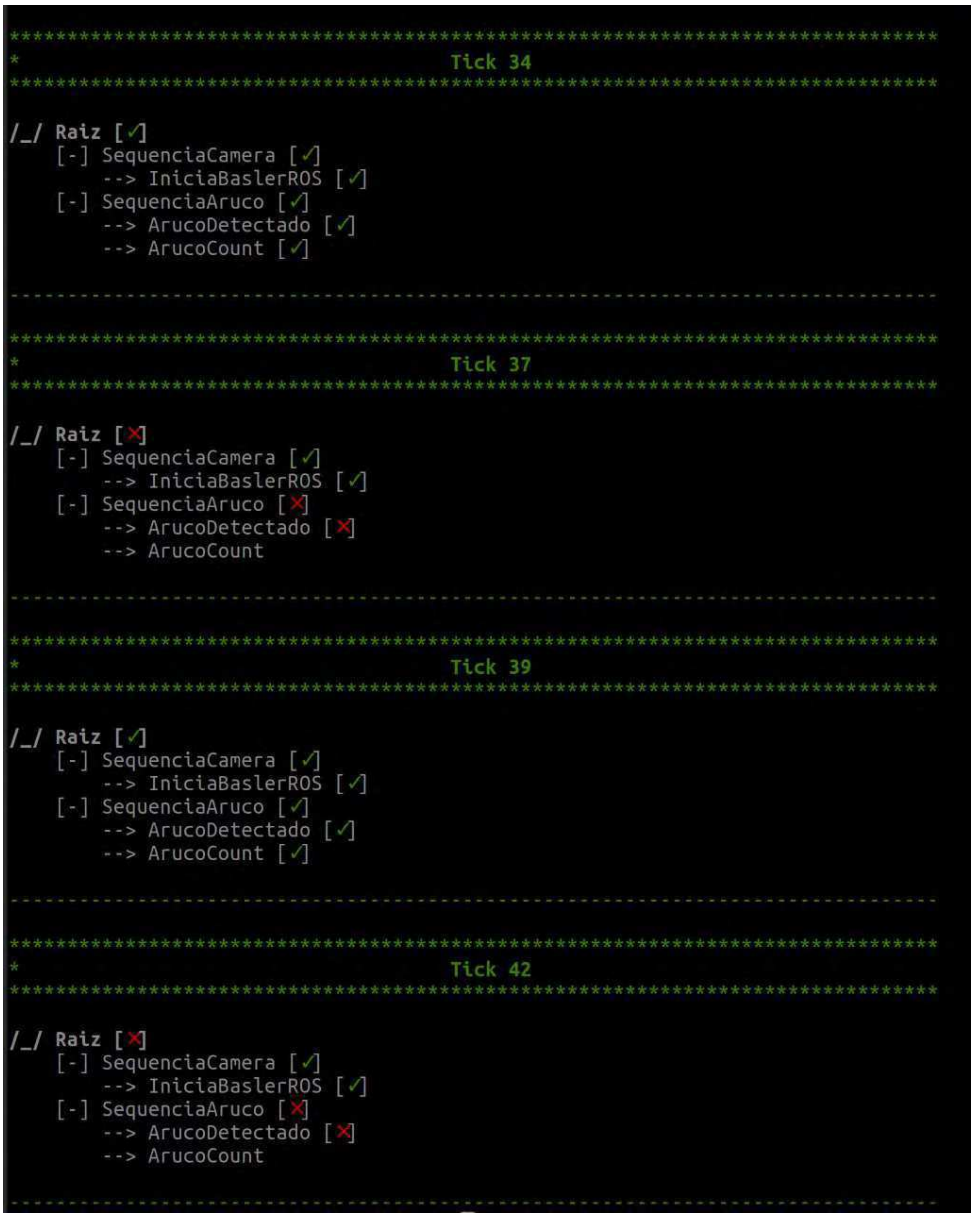


Figura 22 – Quantidade de tick com a velocidade da esteira 50%

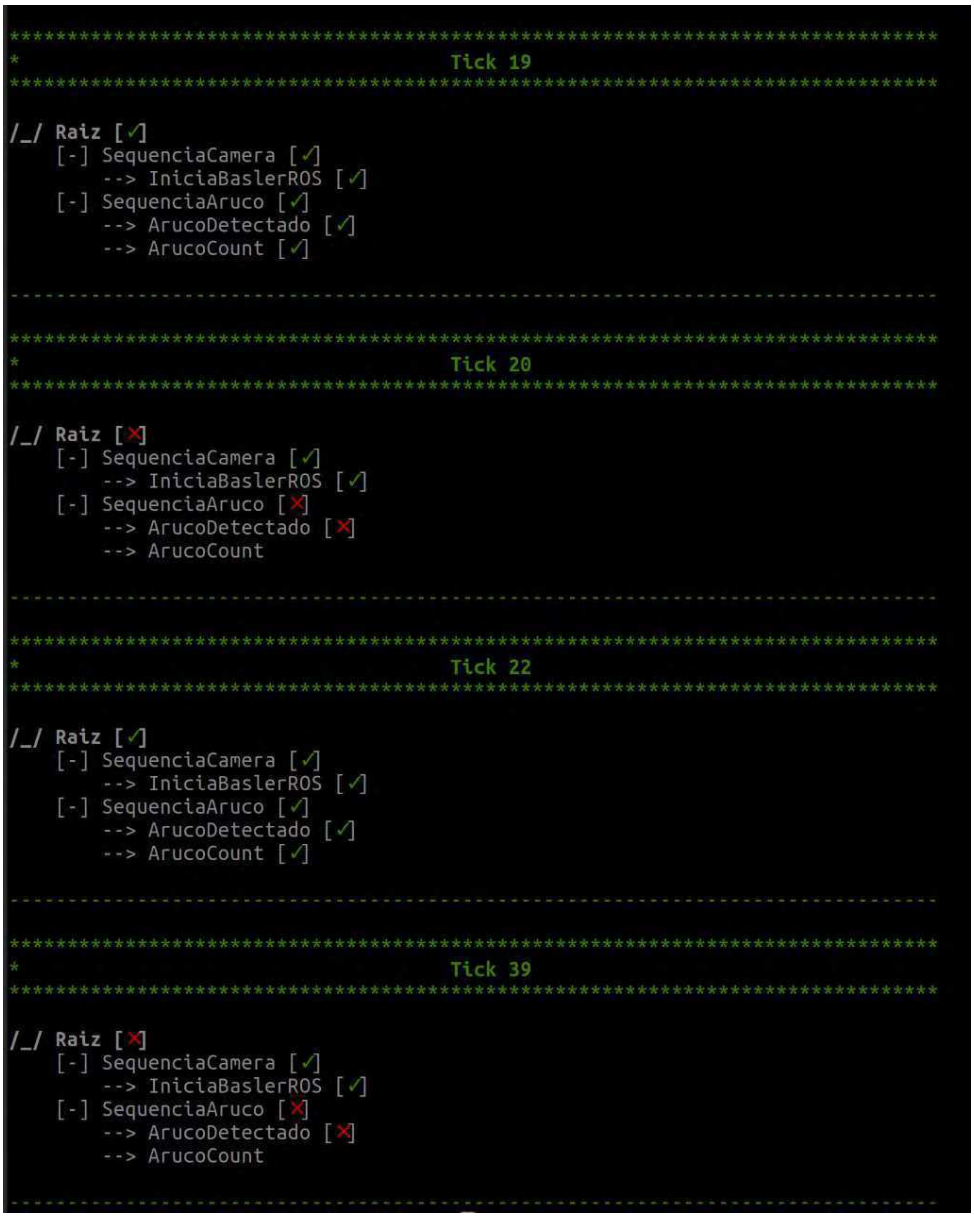


Figura 23 – Quantidade de tick com a velocidade da esteira 75%

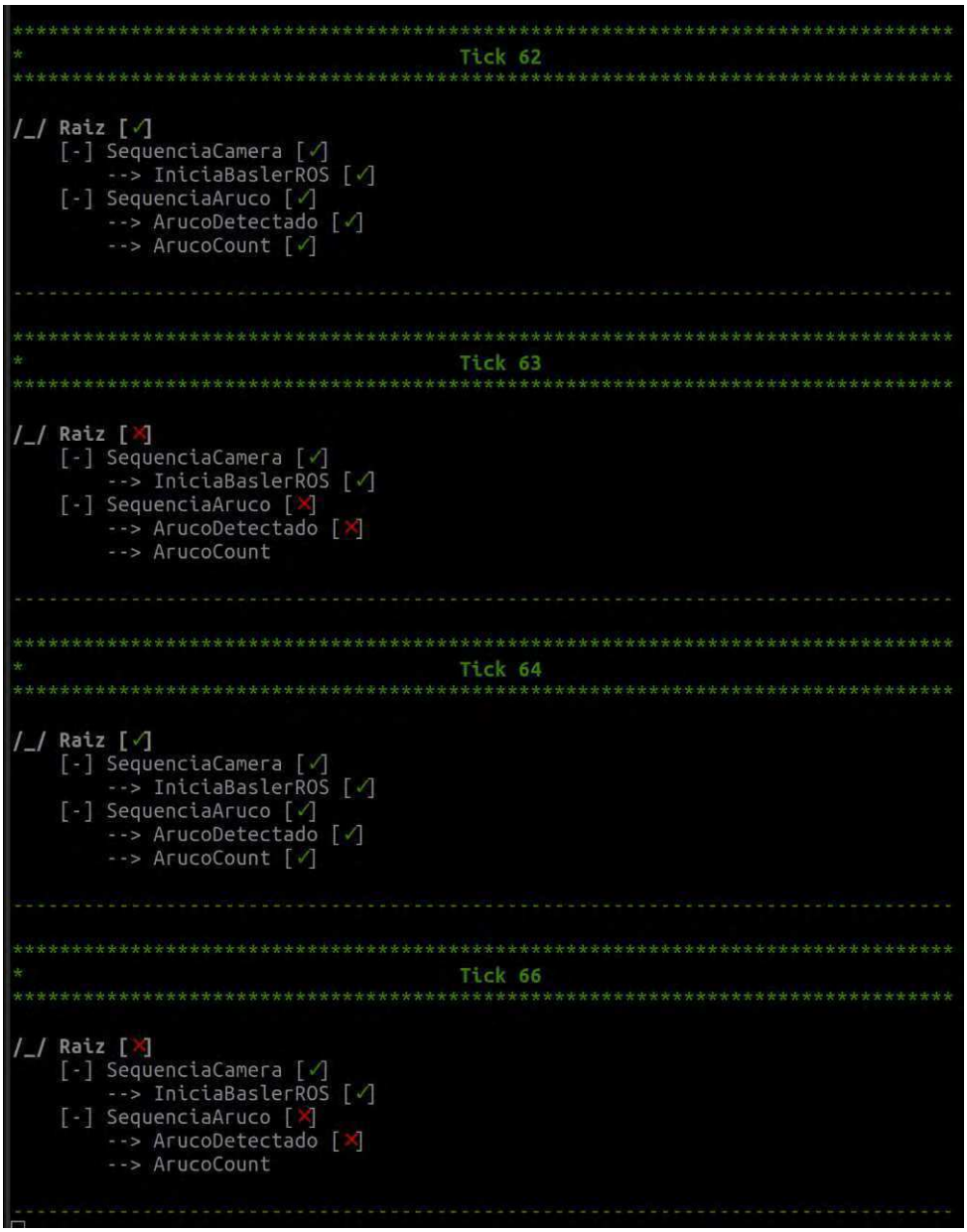


Figura 24 – Quantidade de tick com a velocidade da esteira 100%

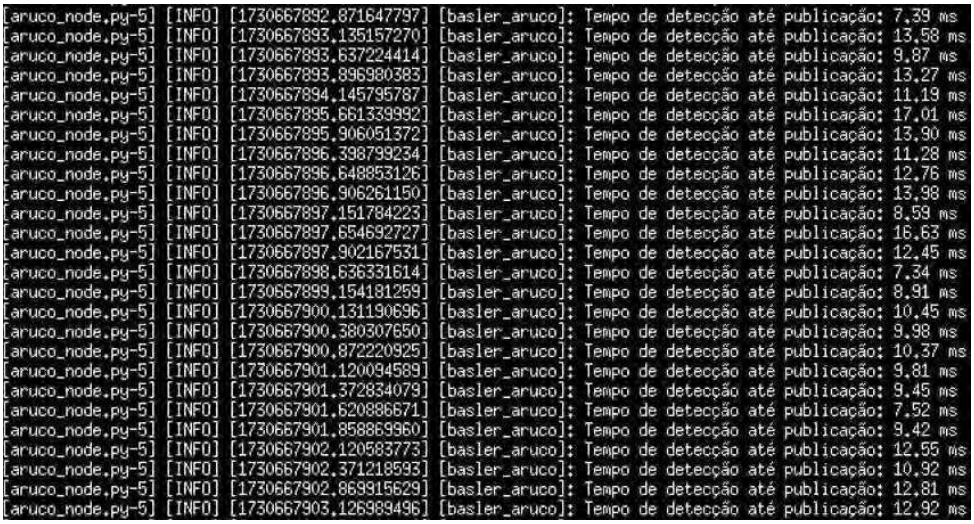


Figura 25 – Tempo de detecção até a publicação de cada ArUco.

6 Conclusão

O trabalho desenvolvido neste projeto resultou em um sistema de visão computacional, utilizando o ROS para identificar e rastrear objetos em movimento em uma esteira industrial. A integração de uma câmera basler com o ROS possibilitou a captura e o processamento de imagens em tempo real, permitindo a detecção e rastreamento de marcadores ArUco. Essa implementação demonstrou que o sistema é capaz de adaptar-se a diferentes velocidades da esteira e operar sob variadas condições operacionais, evidenciando seu potencial para contribuir com a automação e a flexibilidade em ambientes industriais modernos.

6.1 Contribuições do Trabalho

A principal área foco da contribuição neste trabalho é a área de automação industrial com uma solução escalável e modular para a identificação e rastreamento de objetos em uma linha de produção. A implementação com Árvores de Comportamento trouxe uma abordagem estruturada e organizada para gerenciar as tarefas de detecção e manipulação, o que facilita a manutenção e expansão do sistema. Os testes realizados demonstraram que o sistema é capaz de detectar e rastrear todos os objetos, mesmo em altas velocidades da esteira, comprovando sua viabilidade para aplicações industriais.

Além disso, a utilização do ROS como middleware oferece uma plataforma robusta e flexível para gerenciar a comunicação entre os dispositivos, garantindo uma integração entre a câmera Basler e os outros componentes do sistema. A abordagem modular facilita a adição de novas funcionalidades e adaptações para diferentes cenários industriais, reforçando a utilidade e a adaptabilidade do sistema em diversos contextos de produção.

6.2 Limitações e Sugestões para Trabalhos Futuros

Apesar das contribuições significativas, o sistema possui algumas limitações. Uma delas é a dependência de uma frequência de captura de imagem que pode afetar a precisão da detecção em velocidades muito altas. Em velocidades extremas da esteira, o número de poses detectadas de cada marcador ArUco diminui, o que indica uma limitação na taxa de detecção contínua em condições de alta velocidade. Trabalhos futuros podem explorar técnicas de processamento de imagem mais avançadas e de maior taxa de captura para aumentar a precisão, mesmo em condições de alta velocidade.

Outra limitação está relacionada às condições de iluminação. Embora o sistema

funcione bem em ambientes controlados, mudanças drásticas de iluminação podem impactar a detecção dos ArUcos. Estudos futuros poderiam integrar algoritmos de correção de iluminação ou sensores adicionais para melhorar os resultados mesmo em ambientes com iluminação variável.

Sugere-se também a exploração de outras técnicas de visão computacional, como o uso de aprendizado profundo, para aprimorar a identificação e o rastreamento de objetos, permitindo ao sistema reconhecer diferentes tipos de objetos sem a necessidade de marcadores. Além disso, a adição de sensores de profundidade poderia enriquecer a detecção e manipulação de objetos em três dimensões, aumentando ainda mais a aplicabilidade do sistema em contextos industriais complexos.

6.3 Considerações Finais

Em resumo, o trabalho desenvolvido demonstra que um sistema de visão computacional com ROS é uma solução viável e eficiente para o rastreamento de objetos em movimento em uma linha de produção. As contribuições e limitações discutidas servem como base para futuras pesquisas e aprimoramentos, garantindo que o sistema possa evoluir e se adaptar a novas demandas e tecnologias no campo da automação industrial.

Referências

CAñAS, H. et al. Implementing industry 4.0 principles. *Computers & Industrial Engineering*, Elsevier, v. 158, p. 107379, 2021. Citado na página 14.

CHEN, B. et al. Smart factory of industry 4.0: Key technologies, application case, and challenges. In: *Proceedings of the IEEE International Conference on Industry Applications*. [S.l.: s.n.], 2017. v. 4, n. 1, p. 101–102. Citado na página 17.

CSALODI, B. et al. Industry 4.0-driven development of optimization algorithms: A systematic overview. In: *Proceedings of the International Conference on Advanced Engineering Optimization*. [S.l.: s.n.], 2021. v. 3, p. 233–250. Citado na página 16.

EL-IAITHY, R. A.; HUANG, J.; YEH, M. Study on the use of microsoft kinect for robotics applications. *Proceedings of the IEEE Conference*, IEEE, p. 1280–1288, 2012. Citado na página 33.

GHOSON, N. H. et al. Towards remote control of manufacturing machines through robot vision sensors. In: *Proceedings of the International Conference on Engineering Design (ICED23)*. Bordeaux, France: Cambridge University Press, 2023. p. 3601–3608. Citado na página 14.

GUGLIERMO, S. et al. Evaluating behavior trees. In: *Robotics and Autonomous Systems*. [S.l.: s.n.], 2024. v. 178, p. 104714. Citado 3 vezes nas páginas 23, 24 e 25.

HEPPNER, G. Distributed behavior trees for heterogeneous robot teams. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2023. p. 11054–11060. Citado 3 vezes nas páginas 23, 24 e 25.

HUANG, J.-K.; GRIZZLE, J. W. Improvements to target-based 3d lidar to camera calibration. *IEEE Access*, 2017. Funding provided by Toyota Research Institute and NSF Award No. 1808051. Citado na página 26.

LELOWICZ, K. Camera model for lens with strong distortion in automotive application. In: *2019 IEEE International Conference on Instrumentation Control and Automation (ICA)*. [S.l.: s.n.], 2019. p. 314–319. Citado na página 22.

MACENSKI, S. et al. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 2022. Citado 4 vezes nas páginas 17, 18, 20 e 21.

OGREN, P. Behavior trees in robot control systems. In: *CRC Press*. [S.l.: s.n.], 2018. v. 1. Citado na página 24.

ONU, U. N. I. D. O. Industry 4.0 and beyond: Enabling digital transformation and sustainable growth in industry x.0. In: *Annual Report on Industry Transformation*. [S.l.: s.n.], 2023. v. 5, p. 59–78. Citado na página 17.

SHAHRIA, M. T. et al. A comprehensive review of vision-based robotic applications: Current state, components, approaches, barriers, and potential solutions. *Robotics*, v. 11, p. 139, 2022. Citado 3 vezes nas páginas 21, 22 e 23.

SUPRIJANTO; RAKHMAWATI, A.; YULIASTUTI, E. Compact computer vision for black tea quality evaluation based on the black tea particles. In: *2011 2nd International Conference on Instrumentation Control and Automation*. [S.l.: s.n.], 2011. p. 87–91. Citado na página 23.

TEAM, Q. R. Behavior trees for amr robots: Build-upon intelligence. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2023. p. 1154–1160. Citado 2 vezes nas páginas 23 e 24.

WANG, J.; OLSON, E. Apriltag 2: Efficient and robust fiducial detection. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea: IEEE, 2016. Citado 2 vezes nas páginas 26 e 27.

ZEMLA, F. et al. Smart platform for monitoring and control of discrete event system in industry 4.0 concept. In: *Proceedings of the International Conference on Industry 4.0 Technologies*. [S.l.: s.n.], 2023. v. 1, p. 101–112. Citado na página 16.