UNIVERSIDADE FEDERAL DO AMAZONAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE APÓIO À PESQUISA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

# AUTOMATIZAÇÃO E DESENVOLVIMENTO DE UM SISTEMA DE INSPEÇÃO DE PCB UTILIZANDO ROBÓTICA COLABORATIVA E PROCESSAMENTO DE IMAGEM

Alessandra Ribeiro de Menezes

MANAUS-AM
2024

Alessandra Ribeiro de Menezes

# AUTOMATIZAÇÃO E DESENVOLVIMENTO DE UM SISTEMA DE INSPEÇÃO DE PCB UTILIZANDO ROBÓTICA COLABORATIVA E PROCESSAMENTO DE IMAGEM

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como parte dos requisitos necessários para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas na linha de pesquisa Sistemas de Controle e Automação Modernos.

Orientador: Dr. Renan Landau Paiva de Medeiros
Co-orientador: Dr. Iury Valente de Bessa

MANAUS-AM
2024

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Menezes, Alessandra Ribeiro de
M543a Automatização e desenvolvimento de um sistema de inspeção de PCB utilizando robótica colaborativa e processamento de imagem / Alessandra Ribeiro de Menezes . 2024
72 f.: il. color; 31 cm.

Orientador: Renan Landau Paiva de Medeiros
Coorientador: Iury Valente de Bessa
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Amazonas.

1. Indústria 4.0. 2. Robótica Colaborativa. 3. Visão Computacional. 4. PCI - Placa de Circuito Impresso. 5. Sistema de Inspeção. I. Medeiros, Renan Landau Paiva de. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Engenharia Elétrica

**FOLHA DE APROVAÇÃO**

Poder Executivo Ministério da Educação
Universidade Federal do Amazonas
Faculdade de Tecnologia
Programa de Pós-graduação em Engenharia Elétrica

Pós-Graduação em Engenharia Elétrica. Av. General Rodrigo Octávio Jordão Ramos, nº 3.000 - Campus Universitário, Setor Norte - Coroado, Pavilhão do CETELI. Fone/Fax (92) 99271-8954 Ramal:2607. E-mail: ppgee@ufam.edu.br

ALESSANDRA RIBEIRO DE MENEZES

**AUTOMATIZAÇÃO E DESENVOLVIMENTO DE UM SISTEMA DE INSPEÇÃO DE PCB UTILIZANDO ROBÓTICA COLABORATIVA E PROCESSAMENTO DE IMAGEM**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como
requisito parcial para obtenção do título de Mestre
em Engenharia Elétrica na área de concentração
Controle e Automação de Sistemas.

Aprovada em 18 de dezembro de 2024.

BANCA EXAMINADORA
Prof. Dr. Renan Landau Paiva de Medeiros - Presidente
Prof. Dr. Vicente Ferreira de Lucena Junior - Membro Titular 1 - Interno
Prof. Dr. Marcelo de Albuquerque Oliveira - Membro Titular 2 - Externo

Manaus, 09 de dezembro de 2024.

Documento assinado eletronicamente por **Renan Landau Paiva de Medeiros**, **Professor do Magistério Superior**, em 18/12/2024, às 16:24, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Marcelo Albuquerque de Oliveira**, **Professor do Magistério Superior**, em 18/12/2024, às 16:26, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Vicente Ferreira de Lucena Júnior**, **Professor do Magistério Superior**, em 18/12/2024, às 16:35, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2365041** e o código CRC **F6421869**.

Av. Octávio Hamilton Botelho Mourão - Bairro Coroado 1 Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181
CEP 69080-900 Manaus/AM - mestrado_engeletrica@ufam.edu.br

Referência: Processo nº 23105.051851/2024-23 SEI nº 2365041

Criado por 31183646291, versão 2 por 31183646291 em 09/12/2024 15:05:45.

*Dedico este trabalho à minha família, amigos e gatos.*

# Agradecimentos

Agradeço primeiramente a Deus por tornar tudo possível. À minha família, principalmente, aos meus pais e avó por me ensinarem o melhor que podiam. Sou imensamente grata aos amigos e professores da UFAM que acumulei ao longo da graduação e do mestrado. Assim como, aos amigos do trabalho que acompanham minha trajetória de evolução profissional.

Agradeço ao Prof. Renan Landau por todos os ensinamentos, conselhos, orientações e suporte ao longo da graduação e mestrado. E a toda a equipe de professores do PPGEE - UFAM que ajudaram a sedimentar essa formação.

E por fim, agradeço aos meus gatos por serem meus companheiros de pesquisa e escrita durante as madrugadas.

*"A consciência da sua inferioridade é uma prova de humildade sempre agradável a Deus, que leva em conta a intenção caridosa que o anima."*

*(Allan Kardec)*

# Resumo

O avanço das tecnologias habilitadoras do conceito de Indústria 4.0, como a robótica colaborativa e a inteligência artificial (IA), tem contribuído para que a área industrial alcance altos níveis de automação. Um dos grandes desafios da IA aplicada às indústrias é a falta de *datasets* específicos para produtos e modelos variados, essenciais para o treinamento de sistemas inteligentes, sendo o segmento eletroeletrônico um dos mais afetados por essas transformações. Este trabalho propõe uma solução inovadora para automatizar a inspeção de placas de circuito impresso (PCI), incluindo um cadastro inteligente, utilizando técnicas de visão computacional baseadas em aprendizado profundo, como o modelo pré-treinado FastSAM, combinado com Template Matching para compor um sistema eficiente de cadastro automático. A automação deste trabalho foi implementada por meio de um robô colaborativo, capaz de inspecionar os itens em qualquer posição e orientação, sem apresentar riscos aos operadores que realizam outras atividades durante o processo, além de tornar a automação flexível às mudanças no *layout* fabril. O método proposto elimina a necessidade de *datasets* rotulados e amostras defeituosas, permitindo um controle de qualidade mais eficiente e seguro. O sistema atingiu resultados expressivos nas métricas de avaliação, como precisão de 83,77%, revocação de 90,43%, IoU de 76,95%, F1-Score de 86,97% e acurácia de 98,69%, demonstrando alta eficiência na identificação e segmentação de componentes eletrônicos nas PCIs. Como diferencial, a solução demonstra potencial para melhorar significativamente a eficiência dos processos no Pólo Industrial de Manaus (PIM) e em outros cenários industriais.

**Palavras-chave: Indústria 4.0; Robótica Colaborativa; Visão Computacional; PCI; Sistema de Inspeção.**

# Abstract

The advancement of enabling technologies under the concept of Industry 4.0, such as collaborative robotics and artificial intelligence (AI), has contributed to achieving high levels of automation in the industrial sector. One of the major challenges of AI applied to industries is the lack of specific *datasets* for diverse products and models, which are essential for training intelligent systems. The electronics segment is among the most impacted by these transformations. This work proposes an innovative solution to automate the inspection of printed circuit boards (PCBs), including an intelligent registration system, using computer vision techniques based on deep learning, such as the pre-trained FastSAM model combined with Template Matching, to build an efficient automatic registration system. The automation proposed in this work was implemented through a collaborative robot capable of inspecting items in any position and orientation without posing risks to operators performing other tasks during the process, while also ensuring flexibility to adapt to changes in the factory layout. The proposed method eliminates the need for labeled *datasets* or defective samples, enabling more efficient and safer quality control. The system achieved remarkable results in evaluation metrics, such as a precision of 83.77%, recall of 90.43%, IoU of 76.95%, F1-Score of 86.97%, and pixel accuracy of 98.69%, demonstrating high efficiency in identifying and segmenting electronic components on PCBs. As a distinctive feature, the solution shows potential to significantly improve the efficiency of processes in Pólo Industrial de Manaus (PIM) and other industrial scenarios.

**Keywords: Industry 4.0; Collaborative Robotics; Computer Vision; PCB; Inspection System.**

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas e Siglas

| | |
|---|---|
| **PNP** | Pick and Place |
| **PCB** | Printed Circuit Board |
| **IA** | Inteligência Artificial |
| **CQ** | Controle de Qualidade |
| **BGA** | Ball Grid Array |
| **API** | Application Programming Interface |

# Sumário

# Capítulo 1

# Introdução

Os manipuladores robóticos são amplamente utilizados na indústria, devido à sua capacidade de executar diversas atividades, tais como manipulação de objetos, soldagem, pintura, montagem, entre outras aplicações (Han et al., 2020; Francesco and Paolo, 2017). No entanto, existem desafios significativos relacionados à execução dessas tarefas, como a identificação de objetos utilizando sensores ou visão computacional (Pan et al., 2020), manipulação (*grasping*) de objetos com formatos não triviais (Gualtieri and Platt, 2021), planejamento de movimento (Vieira et al., 2019a) e tratamento de incertezas (Vieira et al., 2019b).

De acordo com Nourmohammadi et al. (2024); Doyle-Kent and Kopacek (2021), a Indústria 4.0 busca combinar as competências humanas com as capacidades robóticas. Isso tem impulsionado a inserção de *cobots* no ambiente fabril, devido a diversos benefícios, tais como flexibilidade para adaptação rápida, integração com sistemas digitais, segurança, aumento da produtividade, redução de custos e melhoria da qualidade dos produtos. Robôs colaborativos automatizam tarefas repetitivas e de baixo valor agregado, permitindo que os trabalhadores se concentrem em atividades mais complexas, enquanto garantem um ambiente de trabalho seguro e eficiente.

A colaboração humano-robô no mesmo ambiente de trabalho, assim como a execução de atividades conjuntas, é viabilizada pela implementação dos *cobots*, mas ainda apresenta limitações, causadas pela ausência de percepção e cognição por parte do robô (Villani et al., 2018). Para solucionar essas limitações, é fundamental implementar métodos de controle adaptativos e ajustáveis, capazes de fornecer percepção ao robô, permitindo a tomada de decisões em tempo real sobre como, onde e quando executar as tarefas (Male and Martinez-Hernandez, 2023).

Uma das percepções mais desafiadoras é emular a visão humana, ou seja, prover ao *cobot* uma visão de máquina semelhante à de um humano. Nesse contexto, a visão computacional desempenha um papel essencial nos processos de produção, automatizando tarefas de inspeção e garantindo a qualidade dos produtos (Yang and Kang, 2023).

Voltando-se ao PIM (Pólo Industrial de Manaus), as PCBs constituem uma parte essencial dos equipamentos eletrônicos produzidos, desempenhando um papel crucial na cadeia de produção da indústria eletrônica. A aplicação da visão computacional tem se mostrado fundamental para melhorar a eficiência

da produção, reduzindo o desperdício de materiais e recursos, além de aumentar a qualidade do produto final (Lakhe and Shinghare, 2022).

O segmento eletroeletrônico, no entanto, é um dos que mais sofrem transformações, tanto em seus processos de fabricação quanto no produto final. Além disso, enfrenta-se a problemática da falta de dados anotados realizar a identificação de modelos correspondentes, como por exemplo o treinamento de redes neurais. Empresas desse setor produzem uma vasta gama de modelos, mas frequentemente não possuem pessoal dedicado à tarefa de anotação de dados, especialmente em imagens de placas eletrônicas, que contêm diversos mini e microcomponentes. O cadastro de uma nova placa e a seleção de cada componente torna-se inviável, devido ao alto investimento de tempo e mão de obra necessários.

A partir desse cenário problemático, este trabalho propõe o desenvolvimento de um sistema de visão de máquina composto por segmentação automática em tempo real, baseada em aprendizado *zero-shot* do modelo FastSAM, em conjunto com o Template Matching para buscar padrões ideais em novas placas do mesmo modelo. Essa metodologia proposta elimina a necessidade de gerar *datasets* rotulados para cada modelo de produto, além de dispensar a obtenção de amostras defeituosas.

## 1.1 Estado da Arte

A detecção de objetos-alvo por robôs, utilizando técnicas de visão computacional e aprendizado de máquina, tem sido investigada em diversos estudos com aplicações variadas, desde o setor de saúde até o industrial. Esses estudos abrangem desde o uso de processamento digital de imagens para problemas controlados até o emprego de redes neurais densas com aprendizado profundo.

Em (Aarthi. and Rishma., 2023), foi proposto um método para detecção e segregação automática de resíduos em tempo real, facilitando a tarefa de reciclagem. O trabalho aborda a problemática da identificação de resíduos no meio ambiente, que pode ser um desafio devido à variabilidade de formas, tamanhos e cores dos resíduos. Para resolver este problema, os autores utilizam técnicas de visão computacional para identificar objetos e características geométricas, aplicando a arquitetura Mask RCNN para localizar e classificar os resíduos. Entre os pontos fortes, destaca-se a capacidade de processamento em tempo real; no entanto, a dependência de iluminação uniforme é uma limitação em áreas abertas. Os resultados mostraram que o método alcançou alta precisão na classificação de resíduos, com um impacto potencial positivo na reciclagem automatizada.

Em (Ioller, 2019), foi desenvolvido um robô para separação de garrafas de vidro e plástico. O problema abordado é a dificuldade de separar materiais similares visualmente em ambientes industriais. A solução proposta combina uma CNN leve com extração de características físicas, como densidade e textura, para melhorar a robustez da separação. O principal ponto forte é a integração de aprendizado de máquina e sensores físicos, enquanto a fragilidade está na limitação para novos materiais não treinados. Os resultados indicaram uma precisão média de 95% na separação de garrafas.

Em (Kim et al., 2019), foi proposto um sistema industrial de reciclagem robótica que integra processamento de imagens, planejamento de movimento e classificação de materiais. Este trabalho trata do desafio de automatizar completamente a triagem de materiais em ambientes industriais. Utilizou-se

um modelo LeNet modificado para classificar objetos em papelão, plástico e vidro, enquanto sensores de profundidade ajustam o planejamento de movimento. O ponto forte é a abordagem integrada, mas o custo elevado do sistema é uma desvantagem. Os testes mostraram uma precisão de 92% na classificação de materiais em tempo real.

No setor de inspeção de PCBs, (Wang et al., 2022) propôs um modelo *few-shot* para detecção de defeitos. O problema enfrentado é a dificuldade de treinar redes profundas com poucos dados rotulados. A solução proposta combina módulos de fusão multiescala e mecanismos de atenção para melhorar a capacidade de detecção de pequenos defeitos. Entre os pontos fortes estão a redução da necessidade de dados e a robustez em detectar defeitos raros. A fragilidade é o maior tempo de processamento em comparação com outros métodos. Os experimentos mostraram uma precisão de 97% em conjuntos de dados de PCBs.

Em (Li et al., 2019), foi proposta uma melhoria do YOLOv3 para detecção de componentes eletrônicos em PCBs. O trabalho aborda a limitação do YOLO original em detectar alvos pequenos e complexos. A solução foi ajustar as camadas de saída do YOLO para quatro níveis, aumentando a precisão. Os pontos fortes incluem maior adaptabilidade a diferentes tamanhos de defeitos, enquanto a limitação está no aumento do tempo de treinamento. Os resultados indicaram um mAP de 93,07%.

O artigo (Yang and Kang, 2023) apresentou melhorias no YOLOv7 para inspeção de PCBs. O problema abordado é a detecção eficiente de pequenos componentes e defeitos em superfícies complexas. A solução incluiu a adição de um módulo SwinV2-TDD e mecanismos de atenção, melhorando a precisão sem comprometer a velocidade. Entre os pontos fortes estão a adaptabilidade e o desempenho superior em comparação a modelos anteriores; uma limitação é o maior consumo de memória. O método alcançou um AP de 98,74%.

Em (Wang and Li, 2023), realizou-se uma análise abrangente sobre técnicas de aprendizado profundo aplicadas à detecção de defeitos em PCBs. O trabalho aborda o problema da falta de padronização entre métodos de inspeção. Os autores revisam diferentes arquiteturas e identificam lacunas, como a necessidade de maior generalização. Apesar deste estudo não apresentar implementações, ele se destaca pelas tendências, como o uso de aprendizado por reforço em inspeção.

Em (Khan and Ahmed, 2023), foi investigada a aplicação de redes neurais convolucionais na classificação de defeitos em PCBs. O trabalho aborda a detecção de falhas microscópicas e não uniformes. A solução combina RetinaNet com regularização para melhorar a robustez. Entre os pontos fortes, estão a detecção de pequenos defeitos; como fragilidade, destaca-se a dificuldade em lidar com grandes variações de dados. Os testes indicaram um ganho de 15% na precisão em relação a métodos baseados em SVM.

O trabalho, de (Smith and Zhang, 2022), utilizou redes neurais convolucionais para identificar seis tipos diferentes de defeitos em PCBs. O desafio enfrentado é a diferenciação de falhas visuais que são similares. A solução foi treinar uma CNN especializada em dados aumentados. Entre os pontos fortes estão a alta precisão e a capacidade de detectar defeitos raros; a fragilidade é a necessidade de pré-processamento intensivo. O método alcançou uma precisão de 96%.

Em (Ahmed and Wang, 2023), foi realizada uma análise comparativa de diversas técnicas de aprendizado profundo. O problema explorado foi a escolha de arquiteturas para diferentes tipos de defeitos.

Os autores aplicaram YOLO, Faster R-CNN e SSD, avaliando métricas como precisão e recall. Entre os pontos fortes está a análise detalhada. E como limitação, o estudo não aborda a adaptação dos modelos. Os resultados indicaram que YOLOv7 teve o melhor desempenho global.

Em (Chen and Li, 2023), o algoritmo YOLO foi integrado a um manipulador robótico para guiar visualmente a captura de objetos específicos. O trabalho aborda o problema da necessidade de precisão em ambientes industriais dinâmicos. Para resolver isso, a abordagem combinou redes YOLO com sensores de profundidade, garantindo uma detecção confiável mesmo em condições de iluminação variável. Os resultados mostraram que o modelo teve uma precisão de 90% em tarefas de manipulação de objetos.

O trabalho (Zhang and Liu, 2023) propôs o SC-YOLO, uma arquitetura otimizada para reconhecimento de logotipos em superfícies industriais. O problema tratado foi a detecção de objetos com padrões não uniformes em tempo real. A solução adicionou mecanismos de atenção contextual ao YOLO, melhorando o desempenho em ambientes industriais. Os resultados demonstraram alta precisão de 92% em aplicações de reconhecimento visual.

Em (Wang and Zhao, 2023), introduziu-se o YOLO-CEA, uma abordagem que combina aprimoramento contextual e mecanismos de atenção para melhorar a detecção de defeitos industriais. O problema tratado foi o equilíbrio entre velocidade e precisão em superfícies com alta variabilidade visual. Os experimentos mostraram que o modelo superou abordagens como Faster R-CNN, com uma precisão média (mAP) de 95%.

No artigo (Li and Chen, 2023), o YOLOv6 foi aplicado a detecção de objetos em ambientes industriais. A problemática abordada foi a dificuldade de ajustar arquiteturas YOLO para tarefas específicas com requisitos de alta precisão. A abordagem adaptou camadas convolucionais para otimizar a detecção em superfícies metálicas. O modelo alcançou um aumento de 5% em precisão em comparação com o YOLOv5.

Em (Kumar and Singh, 2024), foi proposta uma extensão do YOLOv8 chamada YOLOv8-TDD, para detecção de defeitos em PCBs. O trabalho abordou o problema de identificar componentes minúsculos e altamente variáveis. A solução incluiu um mecanismo de atenção multiescala para melhorar a precisão. Os resultados mostraram um mAP de 94%, destacando-se em tarefas com componentes pequenos.

Em (Garcia and Lopez, 2023) propôs uma aceleração baseada em FPGA para o YOLOv5, voltada para aplicações em tempo real. O problema enfrentado foi a limitação de hardware convencional em cenários industriais. O FPGA reduziu significativamente o tempo de inferência, mantendo uma precisão média de 91%.

Em (Patel and Sharma, 2023), foi realizado um levantamento sistemático sobre técnicas de aprendizado profundo aplicadas à detecção de defeitos industriais. O trabalho aborda o problema da falta de padronização e da complexidade em adaptar modelos de deep learning a diferentes tipos de defeitos. Para resolver esse desafio, os autores revisaram abordagens baseadas em CNNs, YOLO, Faster R-CNN e transformers, destacando suas vantagens e limitações. Os pontos fortes incluem a abrangência da revisão, cobrindo aplicações em superfícies metálicas, PCBs e produtos têxteis. A principal limitação do estudo é a ausência de implementações práticas. O artigo destaca que modelos como YOLOv7 e Swin-Transformers possuem grande potencial para tarefas em tempo real, mas possui lacunas em cenários de baixa iluminação e ambientes com alta variabilidade.

Em (Wang and Yu, 2024), foi proposta uma versão adaptativa do YOLO para detecção de defeitos em superfícies industriais. O trabalho aborda a necessidade de melhorar a precisão em superfícies com variações de textura e padrões não uniformes. A solução proposta incluiu modificações nas camadas convolucionais e no mecanismo de detecção de múltiplas escalas, adaptando o YOLOv5 para cenários industriais específicos. Entre os pontos fortes estão a robustez em condições de iluminação adversa e a eficiência computacional. Uma limitação é que o modelo requer ajustes finos para diferentes superfícies. Os resultados experimentais mostraram um aumento de 6% no mAP em comparação com o YOLOv5 padrão, alcançando um mAP de 93,2% em aplicações industriais.

Finalmente, em (Nguyen and Pham, 2023) investigou o uso de redes baseadas em transformers para a detecção de defeitos em PCBs. O problema enfrentado foi a dificuldade de detectar falhas minúsculas e irregularidades em superfícies complexas com métodos tradicionais. Os autores propuseram uma arquitetura baseada em Vision Transformers (ViT), que combina autoatenção com camadas convolucionais para capturar padrões locais e globais. O modelo demonstrou pontos fortes em termos de precisão e capacidade de generalização, mas é limitado por sua alta demanda computacional, especialmente durante o treinamento. Os testes em conjuntos de dados de PCBs industriais resultaram em um mAP de 95,8%, destacando-se na detecção de defeitos pequenos e em condições variadas de iluminação.

## 1.2 Motivação e Justificativa

A Indústria 4.0 tem impulsionado a necessidade de integrar competências humanas e robóticas, especialmente em processos de inspeção e controle de qualidade, que são fundamentais para garantir a eficiência e a confiabilidade na fabricação de produtos. No entanto, esses processos enfrentam desafios significativos relacionados à falta de dados anotados, em tempo hábil para o treinamento de redes neurais, além da diversidade de modelos de placas eletrônicas. A ausência de soluções automatizadas e eficientes para inspeção implica em custos elevados, tanto financeiros quanto operacionais, devido à necessidade de mão de obra especializada para anotar dados e realizar inspeções manuais em larga escala, além do tempo dedicado à essa tarefa, por exemplo, para o cadastro de uma nova placa requer a seleção de cada componente manualmente, o que se torna inviável. Além disso, a incapacidade de identificar rapidamente defeitos, inconsistências ou ausências em PCBs pode levar à produção de placas defeituosas, resultando em reprovações e retrabalhos, além do risco do cliente receber a placa com algum dano, o que geraria um problema em potencial, envolvendo a reputação da empresa.

Outro agravante é a diversidade de modelos de placas produzidas, cada uma contendo diversos componentes que exigem inspeções detalhadas e específicas. O alto custo associado ao treinamento de redes neurais personalizadas para cada modelo e a coleta de dados rotulados inviabiliza a adoção de soluções tradicionais de aprendizado de máquina para muitas empresas, especialmente aquelas com altos volumes de produção e necessidade de mundança rápida de *layout* para atender novos produtos. Diante deste contexto, este trabalho propõe uma solução baseada em visão computacional integrada com robótica colaborativa, capaz de otimizar a inspeção de PCBs no Polo Industrial de Manaus (PIM). Essa abordagem visa reduzir custos, aumentar a eficiência e minimizar o impacto da ausência de dados rotulados, tornando

o processo de inspeção mais acessível e eficaz.

## 1.3 Objetivo Geral

Implementar experimentalmente um sistema automático de inspeção de PCBs para processos produtivos industriais, utilizando robótica colaborativa e visão computacional, com o intuito de aplicar inteligência artificial no controle de qualidade do processo produtivo.

## 1.4 Objetivos Específicos

- Desenvolver algoritmos de inteligência artificial focados no processamento de imagens para guiar os movimentos do robô colaborativo;
- Implementar técnicas de segmentação e localização de componentes em PCBs para gerar uma inspeção automática;
- Desenvolver algoritmos de geração de coordenadas da localização das PCBs na área de trabalho delimitada;
- Integrar os algoritmos com robôs colaborativos para automação do processo de inspeção;
- Aplicar métricas quantitativas para avaliar a eficácia da solução proposta.

## 1.5 Organização do Trabalho

O Capítulo 1 apresenta o contexto e a motivação do estudo, enquanto o Capítulo 2 aborda os fundamentos teóricos de visão computacional. O Capítulo 3 compõe os fundamentos teóricos da robótica colaborativa, seguido pelo Capítulo 4, que detalha a metodologia abordada. No Capítulo 5, testes e resultados. Por fim, o Capítulo 5 apresenta as principais conclusões obtidas com o desenvolvimento do estudo, bem como apresenta proposta de trabalhos futuros nos temas relacionados ao trabalho.

# Capítulo 2

# Fundamentação Teórica: Visão Computacional

Este capítulo aborda os conceitos essenciais de visão computacional direcionados para segmentação e detecção de objetos em imagens e vídeos, fornecendo o embasamento necessário para a compreensão e desenvolvimento deste estudo.

## 2.1 Métodos Tradicionais

A visão computacional é definida como o campo dedicado a habilitar as máquinas a interpretar informações visuais para entender e interagir com o mundo ao seu redor. Seu principal objetivo é permitir que computadores analisem e compreendam imagens e vídeos de maneira semelhante aos seres humanos, utilizando algoritmos capazes de detectar, reconhecer e interpretar objetos e cenas complexas em tempo real (Shanmugamani, 2018).

Com essa motivação, surgiram as técnicas clássicas de processamento de imagens, tais como: a filtragem de bordas para detectar contornos (métodos de Sobel e Canny), a extração de características para identificar pontos e padrões (SIFT, SURF e HOG) e a segmentação baseada em limiarização e agrupamento para dividir as imagens em regiões de interesse (Shanmugamani, 2018).

### 2.1.1 Filtros de Borda

De acordo com Gonzalez and Woods (2008); Shanmugamani (2018), os filtros de borda são usados para intensificar e destacar as bordas dos objetos em uma imagem, através das mudanças rápidas de intensidade entre o fundo da imagem e a borda do objeto. Dentre eles, os mais utilizados são:

#### 2.1.1.1 Filtro Sobel

O filtro Sobel calcula os gradientes de intensidade da imagem em direções horizontais ($G_x$) e verticais ($G_y$) para detectar as bordas. É muito aplicado na detecção de contornos onde há variação de brilho entre os objetos da imagem.

Cada uma dessas máscaras convolucionais, $G_x$ e $G_y$, possui dimensões $3\times3$, para calcular os gradientes em uma região local da imagem. As máscaras são definidas como em (2.1).

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{2.1}$$

Vale ressaltar que $G_x$ tem a função de detectar as alterações de intensidade nos pixels de forma horizontal, atribuindo maior peso $\pm2$ às diferenças mais próximas da direção horizontal e menor peso $\pm1$ às diferenças diagonais. Assim como, $G_y$ detecta mudanças de intensidade nos pixels verticalmente, seguindo a mesma ponderação.

Sendo que os valores $+1$, $+2$, $-1$ e $-2$ são os pesos usados na convolução para destacar a contribuição de diferentes vizinhos, ao calcular os gradientes de intensidade. Estes pesos foram escolhidos para realçar a direção do gradiente e suavizar o efeito do ruído na imagem.

- $\pm1$: Representam os pixels mais distantes da borda central na direção do gradiente. Possuem menor peso, visto que estão mais longe do pixel de interesse;
- 0: Indicam o pixel central ($I_{22}$), não contribuem diretamente para o cálculo do gradiente;
- $\pm2$: São os pixels adjacentes diretos ao pixel central na direção do gradiente. Possuem um peso maior ($\pm2$) por serem mais influentes no cálculo do gradiente.

A magnitude do gradiente é calculada pela equação (2.2).

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.2}$$

#### 2.1.1.2 Filtro Canny

Segundo Gonzalez and Woods (2008), O filtro de Canny é amplamente utilizado na detecção de bordas devido à sua capacidade de equilibrar precisão e robustez. Seu funcionamento baseia-se em uma sequência de etapas bem definidas:

1. **Suavização da imagem com filtro Gaussiano:** Para minimizar a influência de ruídos, a imagem é suavizada aplicando-se um filtro Gaussiano. A função Gaussiana bidimensional é dada por:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{2.3}$$

   onde, $\sigma$ representa o desvio padrão, determinando o grau de suavização.

2. **Cálculo dos gradientes:** Após a suavização, calculam-se os gradientes de intensidade nos eixos $x$ e $y$ utilizando operadores diferenciais, como Sobel ou Prewitt. As derivadas parciais são:

$$G_x = \frac{\partial I}{\partial x} \quad \text{e} \quad G_y = \frac{\partial I}{\partial y} \tag{2.4}$$

onde, $I$ é a matriz de intensidade da imagem, representada como uma função bidimensional $I(x, y)$, cujos elementos correspondem aos valores de brilho dos pixels em uma posição específica $(x, y)$. Estes valores geralmente variam entre 0 (preto) e 255 (branco) em imagens de escala de cinza.

A magnitude do gradiente $G$ e a direção $\theta$ são obtidas por:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.5}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{2.6}$$

3. **Supressão de não-máximos:** Esta etapa visa afinar as bordas, eliminando respostas não máximas ao longo da direção do gradiente. Para cada pixel, verifica-se se sua magnitude é máxima em relação aos vizinhos na direção do gradiente. Se não for, o pixel é suprimido.

4. **Limiarização com histerese:** Utilizam-se dois limiares, $T_{\text{alto}}$ e $T_{\text{baixo}}$. Pixels com magnitude acima de $T_{\text{alto}}$ são considerados bordas fortes. Pixels com magnitude entre $T_{\text{baixo}}$ e $T_{\text{alto}}$ são bordas fracas e só são mantidos se conectados a bordas fortes. Este processo assegura a continuidade das bordas e reduz falsas detecções.

A combinação dessas etapas permite que o filtro de Canny detecte bordas de forma precisa, mesmo na presença de ruído, tornando-o ideal para aplicações que requerem um equilíbrio entre sensibilidade e especificidade na detecção de contornos.

### 2.1.2 Extração de Características

Segundo Shanmugamani (2018), a extração de características visa identificar pontos, bordas ou regiões que representem um objeto de maneira única e que possam ser utilizadas para separar o objeto do fundo da imagem, ao extrair descritores robustos às transformações de escala e rotação. As principais técnicas são:

- **SIFT (*Scale-Invariant Feature Transform*):** detecta pontos de interesse em uma imagem que são invariantes à escala, rotação e iluminação. O SIFT cria descritores de características que permitem comparar e combinar objetos em diferentes imagens, sendo muito usado em reconhecimento de objetos e alinhamento de imagens;

- **SURF (*Speeded Up Robust Features*):** versão aprimorada do SIFT, que também é robusta às mudanças de escala e rotação, mas apresenta melhorias em questão de eficiência computacional, permitindo aplicações em tempo real;

- **Histograma de Gradientes Orientados (HOG):** descreve a distribuição dos gradientes de intensidade, faz uma escala das mudanças na luminosidade ao longo da imagem, criando um histograma que caracteriza a forma e os contornos dos objetos. Esta técnica é muito aplicada na detecção de pessoas, onde os gradientes descrevem a silhueta do corpo humano.

### 2.1.3 Segmentação por Limiarização e Agrupamento

O processo de segmentação tem como objetivo dividir a imagem em regiões significativas, agrupando os pixels com características semelhantes, seja por cor ou intensidade. Esta técnica é altamente eficaz em imagens com alto contraste entre o objeto e o fundo da imagem. O processo converte uma imagem em escala de cinza em uma imagem binária e envolve a seleção de um valor de limiar (*threshold*) que separa os pixels da imagem em duas ou mais classes, com base em sua intensidade, destacando o objeto de interesse do fundo. Este processo é regido por um valor de limiar ($T$) (Jähne, 2002; Gonzalez and Woods, 2008), conforme equação (2.7).

$$f(x,y) = \begin{cases} 1, & \text{se } g(x,y) > T \\ 0, & \text{caso contrário} \end{cases} \tag{2.7}$$

onde $g(x,y)$ é o valor de intensidade do pixel na posição $(x,y)$.

Contudo, a segmentação é normalmente utilizada para localizar objetos e limites, dependendo do alvo em questão. Mais precisamente, trata-se do processo de atribuir um rótulo a cada pixel que compõe a imagem processada, garantindo que os pixels com o mesmo rótulo tenham as mesmas características (Shapiro and Stockman, 2001).

A limiarização é o método mais simples e a base para muitos outros métodos de segmentação de imagens. Este método tem como princípio o valor limite para transformar uma imagem em escala de cinza em uma imagem binária. Diversas aplicações na indústria partem deste método, incluindo a entropia máxima, limiar de histograma balanceado, método de Otsu e agrupamento de k-médias (Shapiro and Stockman, 2001; Otsu, 1979).

O método de Otsu é uma técnica popular para determinar automaticamente o valor ideal de $T$, baseado na minimização da variância intraclasse dos pixels do objeto e do fundo Otsu (1979), conforme equação (2.8).

$$\sigma^2(T) = w_1(T)\sigma_1^2(T) + w_2(T)\sigma_2^2(T) \tag{2.8}$$

onde:

- $w_1(T)$ e $w_2(T)$ são as probabilidades do objeto e do fundo, respectivamente.
- $\sigma_1^2(T)$ e $\sigma_2^2(T)$ são as variâncias do objeto e do fundo.

Este método é amplamente utilizado devido à sua simplicidade e eficácia em cenários com boa separação de intensidade entre objeto e fundo.

Já a segmentação por agrupamento é uma técnica que agrupa os pixels com base em características como cor, intensidade ou textura. Segundo Gonzalez and Woods (2008); Shanmugamani (2018), o algoritmo K-means é um método popular de agrupamento, onde um número "K" de *clusters* (grupos) é inicialmente escolhido. Então os pixels são atribuídos aos *clusters* com base em sua similaridade com os centroides destes. A partir disso, os centroides são recalculados e os pixels são reagrupados iterativamente até que os grupos se estabilizem, representando cada um uma região distinta.

Se uma imagem estiver irregularmente iluminada, a primeira coisa a se fazer é otimizar a iluminação da área de trabalho. Caso isso não seja possível, o próximo passo seria identificar a irregularidade do sistema de iluminação e utilizar técnicas de processamento de imagem correspondentes para corrigi-la (Jähne, 2002).

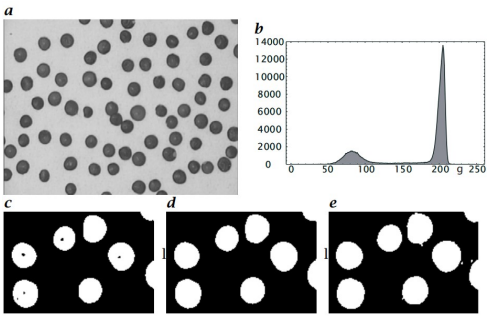



Figura 2.1: Segmentação por limite global. a) Imagem original. b) Histograma. c, d, e) Resultados da segmentação (Jähne, 2002).

Uma grande questão é como definir ou encontrar o limite ideal para o objeto em análise. Na figura 2.1 é mais fácil definir tanto o fundo quanto o objeto por terem valores de cinza bastante uniformes. No caso simples do objeto possuir uma borda simétrica, o limite correto é dado pelo valor médio de cinza entre o fundo e os pixels do objeto. Entretanto, quando a imagem possuir um fundo não uniforme ou se houver objetos com diferentes valores de cinza, esta estratégia falha (Jähne, 2002).

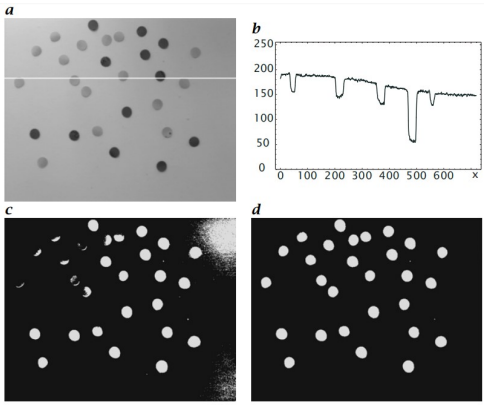



Figura 2.2: Segmentação de uma imagem com iluminação de fundo não homogênea. a) Imagem original. b) Perfil da linha 186 (conforme marcado em "a"). c, d) Resultados da segmentação com um limite global ideal das imagens em "a"antes e depois da imagem ser corrigida para o fundo não homogêneo (Jähne, 2002).

A figura 2.2 apresenta uma imagem com objetos de forma circular, mas com diferente intensidade de cor. Os valores de cinza dos círculos mais brilhantes não formam mais um máximo distinto, como é o caso mostrado na figura 2.1, mas se sobrepõem à ampla distribuição do fundo. Logo, vê-se que o limiar global falha. Mesmo com um limite ideal, parte do fundo nos cantos superior e inferior direitos é segmentada como objetos e os círculos mais brilhantes ainda são segmentados apenas parcialmente. Para corrigir isto,

necessita-se ajustar a iluminação não homogênea para que todos os objetos sejam segmentados (Jähne, 2002).

Para a correção da iluminação, sugere-se assumir que o valor de cinza na imagem é um produto da irradiância não homogênea e da refletividade ou transmissividade do objeto alvo. Além disto, assume-se que se pode tirar uma imagem de referência sem absorver objetos ou com um objeto de refletividade constante. Uma imagem de referência também pode ser computada, quando pequenos objetos são distribuídos aleatoriamente na imagem. Então, basta calcular a imagem média de muitas imagens com os objetos. A iluminação não homogênea pode então ser corrigida dividindo a imagem atual a ser analisada pela imagem de referência (Jähne, 2002).

### 2.1.4 Classes de Segmentação

A segmentação de imagens é uma área importante no campo da visão computacional que ajuda computadores a entenderem não apenas o que está em uma imagem (classificação) ou onde os objetos estão (detecção), mas também os contornos exatos de cada objeto (segmentação). Isto é especialmente útil em robótica e sistemas autônomos, pois permite que os sistemas reconheçam e interajam com o ambiente de forma precisa, identificando objetos, suas formas e posições para realizar tarefas como navegação, manipulação de objetos e tomada de decisões em tempo real. As aplicações variam desde braços robóticos que precisam pegar itens específicos e até veículos autônomos que precisam distinguir entre pedestres, veículos e obstáculos. A segmentação de objetos em imagens, aborda três principais classes, dentre elas estão: segmentação semântica, segmentação de instâncias e segmentação panóptica (Guo et al., 2020).

A segmentação semântica é uma técnica que identifica, pixel a pixel, a classe de cada objeto em uma imagem. Na prática, isso significa que, em uma imagem com várias pessoas, todos os pixels pertencentes a qualquer pessoa serão classificados como "pessoa", enquanto os pixels do fundo serão rotulados como "fundo". Essa abordagem é utilizada para distinguir áreas de interesse, como objetos ou obstáculos, do ambiente, permitindo uma melhor percepção e interação com o espaço ao redor (Guo et al., 2020).

Por outro lado, a segmentação de instâncias vai além da semântica, identificando, pixel a pixel, não apenas a classe do objeto, mas também cada instância individual presente na imagem. Um exemplo prático: em uma imagem com várias pessoas, cada pessoa será segmentada como um objeto único, em vez de serem agrupadas em uma única classe geral, como "pessoa". Uma grande aplicação para esse tipo de segmentação em robótica e sistemas autônomos é a função de distinguir entre múltiplos objetos semelhantes, como identificar e manipular objetos específicos em um ambiente complexo ou monitorar indivíduos em aplicações de segurança e controle (Yi et al., 2019).

Além disso, a segmentação panóptica combina os conceitos da segmentação semântica e de instâncias, oferecendo uma abordagem ainda mais robusta. Assim como a semântica, ela identifica, pixel a pixel, os objetos alvo na imagem, separando o que pertence ou não a uma determinada classe. Ao mesmo tempo, assim como na segmentação de instâncias, a panóptica distingue diferentes instâncias dentro de uma mesma classe, permitindo uma análise detalhada e abrangente do cenário. Isto a torna particularmente útil em situações que exigem alta precisão tanto na identificação das classes, quanto na diferenciação entre instâncias individuais (Kirillov et al., 2019).

Contudo, a segmentação de imagens é classificada em três categorias principais: semântica, de instâncias e panóptica. Cada classe é definida com base em diferentes objetivos e aplicações, desde a rotulação de pixels por classe até a diferenciação entre instâncias individuais de objetos, conforme visto anteriormente. Para tornar mais clara a visualização destas diferenças, tem-se a Tabela 2.1 que fornece uma visão geral dessas classes e suas aplicações típicas.

Tabela 2.1: Comparação entre classes de segmentação (Kirillov et al., 2019).

| **Classe** | **Definição** | **Exemplo de Aplicação** |
|---|---|---|
| Semântica | Rotula todos os pixels de uma classe geral | Separar pedestres de veículos em imagens de tráfego |
| Instâncias | Distingue diferentes instâncias da mesma classe | Contar quantas pessoas estão presentes em uma cena |
| Panóptica | Combina segmentação semântica e de instâncias | Análise detalhada de cenas complexas |

### 2.1.5 Comparação entre Técnicas Clássicas de Visão Computacional

Diante das diversas abordagens tradicionais para segmentação de imagens citadas anteriormente, cada técnica apresenta vantagens específicas dependendo do contexto e da aplicação desejada. A Tabela 2.2 resume os principais métodos, destacando suas características, benefícios e limitações para auxiliar na escolha da abordagem mais adequada.

Tabela 2.2: Comparação entre técnicas clássicas de segmentação (Gonzalez and Woods, 2008).

| **Técnica** | **Princípio** | **Vantagens** | **Desvantagens** |
|---|---|---|---|
| Limiarização | Separação com base em um valor de intensidade | Simplicidade; rápidez | Ineficiente em iluminação não uniforme |
| Agrupamento (K-means) | Agrupamento de pixels com base em similaridade | Boa para padrões de cores | Sensível à inicialização; computacionalmente caro |
| Sobel e Canny | Detecção de bordas usando gradientes | Precisão de bordas; simplicidade | Limitações com ruídos e texturas complexas |

### 2.1.6 Momentos de Imagem

Os momentos de imagem são ferramentas matemáticas utilizadas para caracterizar a forma, posição e orientação de objetos em uma imagem. Eles são amplamente aplicados na visão computacional para o cálculo do centróide (centro de massa geométrico) e da orientação angular do objeto (Gonzalez and Woods, 2008).

#### 2.1.6.1 Momentos Geométricos

Os momentos geométricos de uma imagem $f(x, y)$ de tamanho $M \times N$ são definidos por:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x, y) \tag{2.9}$$

onde $p$ e $q$ são inteiros não negativos que representam a ordem do momento.

**Centróide:** O centróide da área $(cx, cy)$ é calculado com base nos momentos de primeira ordem:

$$cx = \frac{m_{10}}{m_{00}}, \quad cy = \frac{m_{01}}{m_{00}} \tag{2.10}$$

onde $m_{00}$ é a área total da região não nula (soma dos pixels da região).

#### 2.1.6.2 Momentos Centrais

Os momentos centrais são uma extensão dos momentos geométricos, centrados no centróide da área:

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - cx)^p (y - cy)^q f(x, y) \tag{2.11}$$

Eles são utilizados para descrever propriedades invariantes, como orientação.

#### 2.1.6.3 Cálculo da Orientação do Objeto

A orientação de um objeto em uma imagem binária pode ser determinada utilizando os momentos centrais de segunda ordem:

$$\theta = \frac{1}{2} \arctan\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \tag{2.12}$$

onde $\theta$ é o ângulo entre o eixo maior do objeto e o eixo horizontal da imagem.

A orientação angular é uma característica importante para o alinhamento robótico em sistemas de visão, especialmente em tarefas que exigem manipulação precisa do objeto, como o *pick-and-place*.

## 2.2 Aprendizado Profundo em Imagens

O aprendizado profundo tem revolucionado diversas áreas da inteligência artificial, principalmente no processamento de imagens e vídeos. Aplicações como detecção, classificação e segmentação de objetos alcançaram novos patamares de precisão, viabilizando soluções inovadoras em áreas como saúde, robótica, transporte autônomo e vigilância. Entretanto, esses avanços ainda enfrentam desafios significativos, especialmente relacionados à necessidade de grandes volumes de dados rotulados e à generalização de modelos para novas classes ou cenários complexos. Métodos emergentes, como *Few-Shot Learning* e *Zero-Shot Learning*, têm ganhado destaque ao buscar superar estas limitações, oferecendo abordagens eficientes para treinar modelos com poucos ou nenhum exemplo de uma nova categoria (Wang et al., 2022).

### 2.2.1 Aprendizado de Máquina Tradicional e Suas Limitações

O aprendizado de máquina tradicional tem sido amplamente utilizado para resolver uma variedade de problemas, desde reconhecimento de padrões até previsão de séries temporais. No entanto, esses modelos geralmente dependem de grandes conjuntos de dados rotulados para alcançar um desempenho satisfatório (Bishop and Nasrabadi, 2006). A coleta e rotulação de grandes quantidades de dados podem ser caras e demoradas, além de, em alguns casos, inviáveis devido à escassez de dados disponíveis ou à natureza dinâmica das classes de interesse.

Modelos tradicionais operam com uma lista fixa de categorias, o que significa que a introdução de novas classes requer o treinamento de um novo modelo ou o ajuste do modelo existente. Além disso, esses modelos tendem a ser menos eficazes ao lidar com classes com poucas amostras, pois não generalizam bem em cenários de dados limitados.

### 2.2.2 Aprendizado por Transferência (*Transfer Learning*)

O aprendizado por transferência surge como uma solução para mitigar as limitações dos modelos tradicionais. Essa abordagem aproveita o conhecimento adquirido por um modelo em uma tarefa ou domínio específico e o aplica a uma tarefa ou domínio relacionado (Pan and Yang, 2009). Por exemplo, um modelo pré-treinado em um grande conjunto de dados como o ImageNet (Deng et al., 2009) pode ser adaptado para uma tarefa de classificação específica com um conjunto de dados menor, ajustando apenas as camadas finais ou refinando todo o modelo.

O aprendizado por transferência tem se mostrado eficaz em reduzir o tempo de treinamento e a necessidade de grandes conjuntos de dados rotulados, permitindo que modelos alcancem alto desempenho mesmo em tarefas com dados limitados (Yosinski et al., 2014).

### 2.2.3 Aprendizado *Zero-Shot* e *One-Shot*

Para superar ainda mais as limitações associadas à dependência de dados rotulados, surgiram abordagens como o aprendizado *Zero-Shot* e *One-Shot*.

#### 2.2.3.1 Aprendizado *Zero-Shot*

O aprendizado *Zero-Shot* permite que modelos reconheçam classes não vistas anteriormente durante o treinamento, baseando-se em informações auxiliares como descrições textuais, atributos ou relações semânticas (Lampert et al., 2009). Dessa forma, o modelo pode inferir características de novas classes e realizar previsões sem exemplos diretos dessas classes.

#### 2.2.3.2 Aprendizado *One-Shot*

O aprendizado *One-Shot* é uma abordagem em que o modelo aprende a reconhecer novas classes com apenas uma única amostra de treinamento por classe (Fei-Fei et al., 2006). Isso é particularmente útil em aplicações onde a coleta de dados é difícil ou cara. Técnicas como redes siamesas e redes de

correspondência são comumente usadas para medir a similaridade entre amostras e auxiliar na classificação com base em poucos exemplos (Koch et al., 2015).

### 2.2.4 Meta-Aprendizado (*Meta-Learning*)

O meta-aprendizado é uma abordagem que capacita modelos a aprenderem novas tarefas rapidamente usando experiências anteriores (Vilalta and Drissi, 2002). Em vez de focar em uma única tarefa, o modelo aprende a partir de uma variedade de tarefas diferentes, extraindo meta-conhecimentos que podem ser aplicados para acelerar o aprendizado em novas tarefas com dados limitados.

Uma das técnicas mais conhecidas em meta-aprendizado é o MAML (*Model-Agnostic Meta-Learning*), que procura encontrar um bom ponto de partida nos parâmetros do modelo, a partir do qual poucas iterações de treinamento em uma nova tarefa levarão a um bom desempenho (Finn et al., 2017).

### 2.2.5 Aplicações Práticas do *Few-Shot Learning*

O *Few-Shot Learning* tem encontrado aplicações em diversas áreas:

- **Reconhecimento de Imagens**: Permite que sistemas identifiquem novas classes de objetos com base em poucas imagens de exemplo, útil em reconhecimento de espécies raras ou novos produtos (Wang et al., 2020);
- **Processamento de Linguagem Natural**: Facilita a adaptação de modelos de linguagem a novos domínios ou tarefas com poucos exemplos, como tradução de idiomas com recursos limitados;
- **Robótica**: Auxilia robôs a aprender novas habilidades ou adaptar-se a novos ambientes com mínimo treinamento, essencial para operações em ambientes desconhecidos (Duan et al., 2017).

### 2.2.6 Desafios e Limitações do *Few-Shot Learning*

Apesar dos avanços, o *Few-Shot Learning* enfrenta desafios significativos:

- **Generalização Limitada**: Modelos podem não generalizar bem para classes ou domínios muito diferentes dos vistos durante o meta-treinamento;
- **Sensibilidade a Variações**: Variações sutis nas amostras de treinamento podem levar a desempenho inconsistente, devido à falta de dados para capturar a diversidade da classe (Hariharan and Girshick, 2017);
- **Overfitting**: Há risco de sobreajuste às poucas amostras disponíveis, comprometendo a capacidade de generalização do modelo (Yin et al., 2020).

### 2.2.7 Evolução para Modelos de Segmentação Generalistas

Com os avanços no *Few-Shot Learning*, surgiu o interesse em desenvolver modelos capazes de segmentar objetos em imagens sem a necessidade de treinamento específico para cada classe. Esses modelos de

segmentação generalistas visam identificar e segmentar uma ampla variedade de objetos, mesmo aqueles não vistos durante o treinamento, combinando técnicas de aprendizado com poucas amostras e aprendizado não supervisionado (Hu et al., 2018).

O *Segment Anything Model* (SAM) e o FastSAM são exemplos de modelos que representam essa evolução, permitindo segmentar qualquer objeto em uma imagem com base em entradas mínimas, como pontos ou caixas delimitadoras, sem treinamento adicional (Kirillov et al., 2023; Zhao et al., 2023).

### 2.2.8 Modelos *Few-Shot*

Uma das principais limitações dos modelos tradicionais de segmentação de imagens é a dependência de grandes *datasets* categorizados, que são caros e demorados para produzir, especialmente devido ao processo de rotulação manual. Além disso, estes modelos operam com uma lista fixa de categorias, exigindo o treinamento de um novo modelo para cada classe adicional. O *Few-Shot Learning* surge como uma abordagem promissora para contornar essas restrições. Ele permite que os modelos aprendam a partir de um número extremamente reduzido de amostras, explorando transferências de conhecimento entre classes conhecidas e novas categorias. Por meio de técnicas de aprendizado profundo, estes métodos utilizam informações auxiliares, como *embeddings* de características, para associar classes previamente observadas com aquelas que ainda não foram vistas, viabilizando uma generalização eficaz mesmo em cenários com dados limitados (Xian et al., 2020; Wang et al., 2022).

#### 2.2.8.1 SAM

O *Segment Anything Model* (SAM) é uma inovação projetada para prever máscaras de segmentação para qualquer objeto de interesse, dada uma imagem de entrada. Desenvolvido pela equipe da *Meta AI Research, FAIR*, o SAM está disponível sob a licença pública *Apache 2.0 license* (Kirillov et al., 2023).

Segundo Kirillov et al. (2023), lançado em 2023, o SAM introduz uma nova abordagem para segmentação, acompanhada pelo maior conjunto de dados já criado: mais de 1 bilhão de máscaras aplicadas a 11 milhões de imagens licenciadas. O modelo foi projetado para ser altamente eficiente e se enquadra na categoria de *Zero-Shot Learning*, possibilitando a segmentação de objetos em novas imagens sem a necessidade de treinamento adicional. Além de sua escalabilidade, o SAM visa acelerar pesquisas em visão computacional, promovendo avanços em modelos baseados em aprendizado profundo.

Figura 2.3: Imagens segmentadas utilizando SAM (Kirillov et al., 2023)

Conforme Kirillov et al. (2023), a arquitetura do SAM é composta por 3 etapas:

- *Vision Encoder*: um módulo de codificação de imagens, baseado em VIT. Tem a tarefa de calcular os *embeddings* da imagem usando a atenção nos trechos da imagem, aplicando também a incorporação posicional relativa na segmentação da imagem;
- *Prompt Encoder*: etapa responsável pela geração dos *embeddings* para pontos e caixas delimitadoras. Vale ressaltar que esse termo de *embeddings* é originado do PNL, processamento de linguagem natural;
- *Mask Decoder*: módulo composto por um *transformer* bidirecional que realiza a atenção cruzada entre a incorporação da imagem e a incorporação do ponto e vice-versa. Com base nisto, o modelo prevê as máscaras de saída.

O modelo atribui uma pontuação na saída, que se refere a uma medida de confiança associada à máscara gerada para o objeto segmentado na imagem, representada por *score*. Este valor de *score* indica o quão certo o modelo está de que a máscara é correspondente ao objeto segmentado. A arquitetura pode ser observada na Figura 2.4.

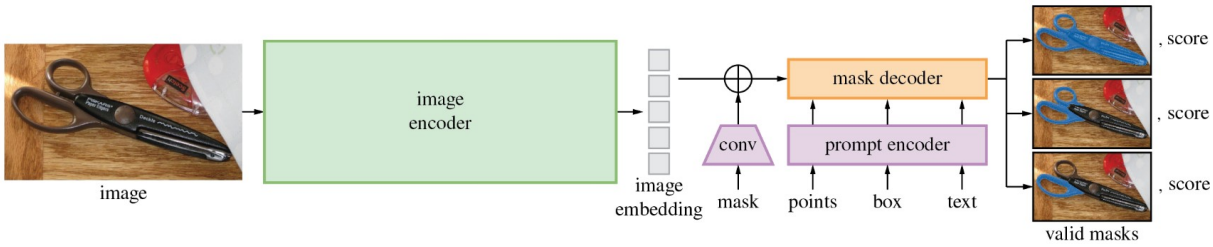



Figura 2.4: Arquitetura do SAM versão ViT-H (Kirillov et al., 2023)

#### 2.2.8.2 FastSAM

De acordo com Zhao et al. (2023), o modelo FastSAM foi desenvolvido para superar as limitações do SAM, principalmente quanto à sua alta demanda computacional. Conforme ilustrado na Figura 2.5, a

arquitetura do FastSAM organiza o processo de segmentação em duas etapas principais: a segmentação de todas as instâncias e a seleção orientada por *prompt*.

Na primeira etapa, o modelo utiliza o YOLOv8-seg para gerar máscaras de segmentação para todas as instâncias presentes na imagem. Essa etapa inclui extração de recursos, geração de protótipos e coeficientes de máscara, permitindo segmentação eficiente e em tempo real Zhao et al. (2023).

Na segunda etapa, o modelo permite ao usuário especificar regiões de interesse, refinando os resultados de acordo com a solicitação, que podem ser: pontos específicos, caixas delimitadoras (*bounding boxes*) ou textos. A partir disso, o modelo usa operações de similaridade ou sobreposição (IoU) para segmentar objetos Zhao et al. (2023).

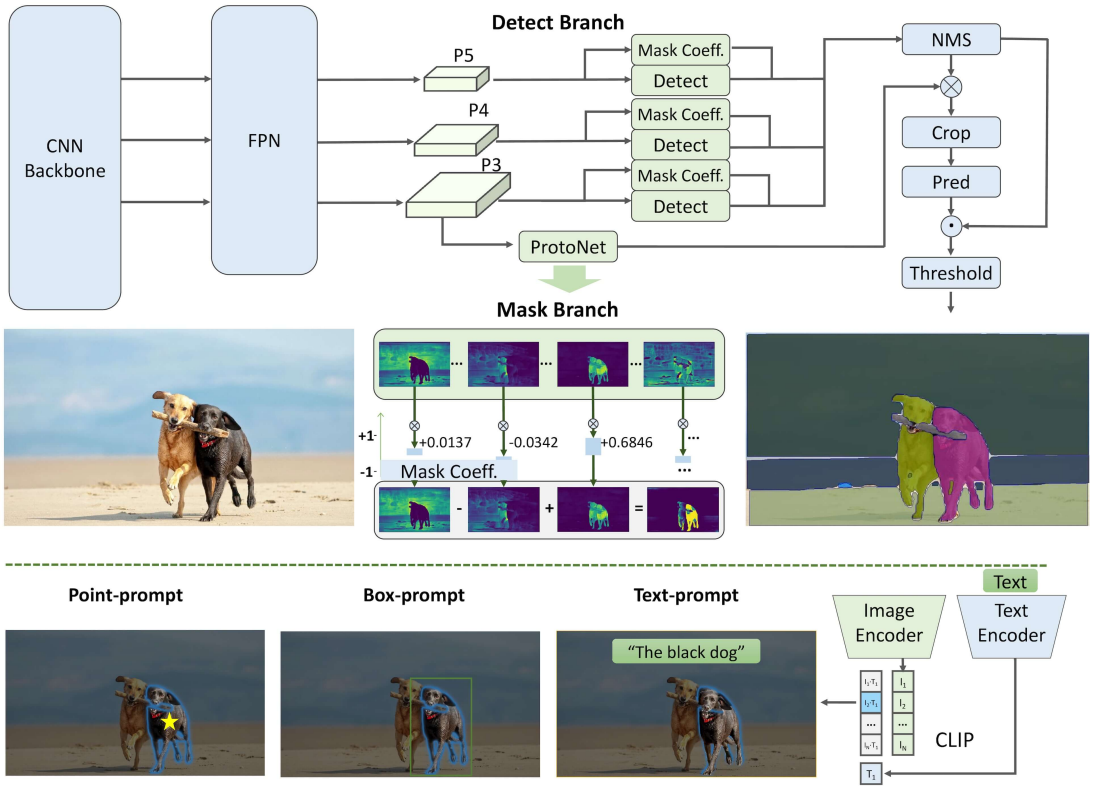



Figura 2.5: Arquitetura do Fast-SAM (Zhao et al., 2023)

Conforme a Figura 2.5, o FastSAM é baseado no YOLOv8-seg, uma versão estendida do YOLOv8 para segmentação de instâncias, que atua na combinação entre detecção de instâncias e segmentação. A arquitetura inclui:

- **Rede *backbone* e FPN (*Feature Pyramid Network*):** O YOLOv8-seg extrai os recursos multiescala da imagem de entrada utilizando uma rede backbone e uma FPN. O *backbone* extrai as características básicas de diferentes níveis da imagem. E a FPN combina informações de diferentes escalas para melhorar a detecção e segmentação de objetos de tamanhos variados;
- **Ramos de detecção e segmentação paralelos:** Os ramos de detecção e segmentação são executados em paralelo. O ramo de detecção gera as classes e caixas delimitadoras para os objetos detectados, enquanto o de segmentação gera as máscaras para os objetos detectados;

- **Prototipagem e Coeficientes de Máscara:** A segmentação utiliza prototipagem de máscaras, baseada no método YOLACT, onde os protótipos de máscara são uma coleção de representações gerais de máscaras, composta normalmente por 32 protótipos. E os coeficientes de máscara são os pesos que combinam os protótipos para gerar as máscaras específicas para cada objeto;

- **Seleção guiada por *prompts*:** Após a segmentação inicial, o modelo refina os resultados com base nos *prompts* fornecidos pelo usuário. Então ocorre a identificação de objetos com base em pontos de interesse, caixas delimitadoras ou textos utilizando métodos como IoU e *embeddings* CLIP para associar o texto às máscaras geradas.

Segundo Zhao et al. (2023), o FastSAM em comparação com o SAM tem seus pontos fontes e algumas limitações, dentre elas, destacam-se:

**Pontos fortes:**

- **Menor custo computacional:** O FastSAM substitui os modelos *Transformers* por uma arquitetura baseada em CNNs, reduzindo significativamente a demanda de hardware;

- **Velocidade superior:** Até 50 vezes mais rápido que o SAM, permitindo aplicações em tempo real;

- **Adaptação eficiente:** Mantém desempenho competitivo mesmo sendo treinado com apenas 2% do conjunto de dados SA-1B.

**Limitações:**

- **Precisão inferior em objetos pequenos:** Máscaras geradas para objetos pequenos podem apresentar baixa qualidade;

- **Artefatos em bordas:** Máscaras de objetos grandes podem ter falhas nas bordas, devido às limitações do método YOLACT;

- **Segmentação baseada em texto:** Apesar de funcional, a integração com *prompts* textuais é mais lenta e menos eficiente do que no SAM.

### 2.2.9 Comparação entre SAM e FastSAM

A análise das arquiteturas SAM e FastSAM evidencia a evolução dos modelos de segmentação, com cada um oferecendo soluções adaptadas a diferentes cenários e recursos computacionais. A Tabela 2.3 apresenta uma visão consolidada das diferenças e semelhanças entre ambos, destacando os pontos fortes e limitações de cada abordagem.

Tabela 2.3: Comparação entre SAM e FastSAM (Kirillov et al., 2023; Zhao et al., 2023).

| **Critério** | **SAM** | **FastSAM** |
|---|---|---|
| Modelo Base | Transformer (ViT-H) | YOLOv8-seg |
| Desempenho | Alta precisão | Alta eficiência em tempo real |
| Custo Computacional | Elevado | Reduzido |
| Treinamento | Conjunto de dados SA-1B completo | Apenas 2% do conjunto de dados SA-1B |
| Limitações | Necessidade de alto poder computacional | Qualidade inferior em objetos pequenos e bordas grandes |

## 2.3 Localização de Objetos

### 2.3.1 *Template Matching*

O *Template Matching* é uma técnica clássica de visão computacional utilizada para identificar regiões de uma imagem que correspondam a um modelo pré-estabelecido, (*template*). Conforme Brunelli (2009), esta abordagem baseia-se na comparação direta entre o *template* e segmentos da imagem-alvo, buscando maximizar a similaridade ou minimizar a diferença entre eles. Ela tem sido amplamente aplicada em tarefas como reconhecimento de caracteres, detecção de objetos e inspeção industrial, devido à sua simplicidade e eficácia em cenários controlados e fixos.

A técnica também se beneficia de estudos teóricos sobre correspondência de padrões na percepção visual, influenciados por conceitos que exploram a organização espacial e estrutural de formas tridimensionais. Embora o *Template Matching* seja voltado para padrões bidimensionais, ele compartilha princípios que envolvem a comparação entre representações conhecidas e novos dados (Marr and Nishihara, 1978).

**Principais Pontos Fortes** De acordo com Brunelli (2009), o *Template Matching* apresenta as seguintes vantagens:

- **Facilidade de Implementação:** A abordagem é direta, requer apenas uma amostra como base para ser o *template* e uma métrica de similaridade, como correlação cruzada ou diferença quadrática média (SSD). Não exige treinamento de modelos, sendo ideal para aplicações com poucos dados ou requisitos simples;
- **Aplicabilidade em Cenários Controlados:** Funciona bem em ambientes onde as condições, como escala, rotação e iluminação, são consistentes, por exemplo: inspeção de peças industriais ou análise de imagens médicas;
- **Flexibilidade de Configuração:** Pode ser adaptada utilizando diferentes métricas de similaridade, técnicas de pré-processamento ou abordagens baseadas em gradientes;
- **Baixa Dependência de Dados Rotulados:** Por ser uma técnica não supervisionada, não requer grandes volumes de dados anotados, tornando-se viável para problemas específicos.

**Principais Limitações:**

Apesar das vantagens citadas anteriormente, o *Template Matching* apresenta desafios, acordados em Brunelli (2009):

- **Sensibilidade a Transformações:** A técnica tradicional é limitada a mudanças de escala, rotação ou perspectiva, comprometendo sua eficácia em cenários com variações significativas;
- **Alto Custo Computacional:** O processo de correspondência em imagens grandes ou complexas pode ser computacionalmente intensivo, por exemplo, análises multiescala ou multiorientação de imagens;
- **Dependência de Condições Ideais:** É menos robusto a ruídos, sobreposições ou condições de iluminação variáveis, o que pode levar a falsos positivos ou negativos;
- **Restrição em Ambientes Complexos:** Métodos modernos, como redes neurais convolucionais (CNNs), frequentemente superam o *Template Matching* em cenários dinâmicos, devido à sua capacidade de generalizar melhor para novos dados.

**Evolução e Integração com Métodos Modernos**

Avanços recentes têm buscado mitigar as limitações do *Template Matching* por meio de sua integração com técnicas modernas Brunelli (2009). Por exemplo:

- Combinação com métodos baseados em aprendizado profundo para geração de *templates* mais adaptáveis.
- Uso de pré-processamento avançado para normalizar escala e rotação.
- Estratégias híbridas que utilizam o *Template Matching* como uma etapa inicial em pipelines mais complexos.

## 2.4 Conclusão

Este capítulo apresentou a fundamentação teórica sobre visão computacional como foco em segmentação de imagens e localização de objetos em imagens. O próximo capítulo abordará os fundamentos da robótica colaborativa, na qual será aplicada a visão computacional deste trabalho.

# Capítulo 3

# Fundamentação Teórica: Robótica Colaborativa

Neste capítulo, serão abordados os fundamentos da robótica colaborativa, com ênfase no robô colaborativo (*cobot*) da *Universal Robots* utilizado neste trabalho. Também será explorada a interação humano-robô no contexto industrial, destacando suas aplicações e benefícios.

## 3.1 Robótica Colaborativa

A robótica colaborativa refere-se à interação direta entre humanos e robôs em ambientes compartilhados de trabalho. Diferentemente dos robôs industriais tradicionais, que operam em áreas isoladas por questões de segurança, os robôs colaborativos (*cobots*) são projetados para trabalhar lado a lado com operadores humanos, priorizando a segurança, adaptabilidade e facilidade de integração (Lucci et al., 2022; Koszulinski et al., 2022).

O conceito de robôs colaborativos surgiu na década de 1990, com o objetivo de criar máquinas capazes de interagir diretamente com humanos. No entanto, foi em 2008 que a *Universal Robots* lançou o UR5, considerado o primeiro robô colaborativo comercialmente viável, marcando um ponto de inflexão na indústria (Universal Robots, 2024b).

A robótica colaborativa tem sido amplamente estudada devido aos seus benefícios e inovações:

- **Benefícios:**
  - **Aumento da Produtividade e Flexibilidade:** Os *cobots* aumentam a produtividade ao realizar tarefas repetitivas de forma eficiente, garantindo flexibilidade para se adaptar a diferentes processos industriais (Baratta et al., 2023; Lucci et al., 2022);
  - **Redução da Carga Cognitiva e Melhoria da Usabilidade:** Reduzem o esforço mental dos operadores, tornando o ambiente de trabalho mais amigável e eficiente (Fournier et al., 2023);

  - **Diminuição de Erros:** Minimiza erros humanos e otimiza a precisão na execução de tarefas (Baratta et al., 2023; Neves et al., 2024).

- **Inovações:**

  - **Sistemas de Captura de Movimento Óptico Vestíveis:** Tecnologias como o CobotGear permitem uma interação mais precisa entre humanos e robôs, aumentando a eficiência em processos complexos (Heredia et al., 2020);
  - **Aprendizado Profundo em Robótica:** O uso de técnicas de aprendizado profundo permite que *cobots* reconheçam padrões e se adaptem a ambientes dinâmicos (Neves et al., 2024);
  - **Indústria 4.0:** Os *cobots* desempenham um papel crucial na Indústria 4.0, integrando dados e automatizando fluxos de trabalho (Pereira and Simonetto, 2021).

### 3.1.1 Colaboração Humano-Robô

A colaboração humano-robô ocorre quando um operador humano interage com o robô durante a execução de atividades de cooperação. Essa interação pode ser dinâmica e ocupar o mesmo espaço de trabalho, seja de forma direta ou indireta (Baratta et al., 2023).

Embora essa forma de colaboração seja estudada desde os primórdios das revoluções industriais, foi na Quarta Revolução Industrial que o conceito ganhou força, encontrando aplicações em diversos setores, como o industrial e o médico. Essa expansão pode ser ilustrada por dados de mercado, que projetam que o segmento de robôs colaborativos (*cobots*) atingirá a marca de 2 bilhões de dólares até 2030 (Baratta et al., 2023).

Para uma colaboração humano-robô eficiente, é essencial garantir cenários colaborativos que sejam sistematicamente otimizados e seguros para os humanos. Nesse contexto, os *cobots* desempenham um papel fundamental, pois são projetados para executar uma variedade de tarefas em conjunto com operadores humanos. O avanço desses robôs foi impulsionado pelos recentes desenvolvimentos tecnológicos promovidos pela Quarta Revolução Industrial, como sensores táteis, sistemas de visão computacional e algoritmos de aprendizado profundo (Ajoudani et al., 2018; Matheson et al., 2019).

Entre as inovações mais significativas está a utilização de sistemas de captura de movimento óptico vestíveis, que permitem interações mais naturais e precisas entre humanos e robôs em ambientes industriais (Heredia et al., 2020). Além disso, o aprendizado profundo tem sido amplamente utilizado para melhorar a eficiência dos *cobots*, otimizando o reconhecimento de objetos e a execução de tarefas em cenários dinâmicos e desestruturados (Neves et al., 2024; Javaid et al., 2022).

## 3.2 Robótica Colaborativa na Indústria

A robótica colaborativa se destaca na indústria por sua capacidade de aumentar a produtividade, promover a ergonomia e garantir elevados padrões de qualidade (Colgate et al., 1996; Guerin et al., 2015). No desempenho de atividades industriais, identificam-se quatro estados de colaboração (Müller et al., 2017):

- **Coexistência:** O robô e o operador compartilham o mesmo espaço, mas não interagem diretamente;
- **Sincronia:** Trabalham no mesmo ambiente, mas em tempos distintos;
- **Cooperação:** Executam atividades diferentes, ao mesmo tempo, no mesmo espaço;
- **Colaboração:** Trabalham juntos, de forma que as ações de um influenciam diretamente as do outro.

Com base nessas definições, pode-se contextualizar que a colaboração humano-robô é estabelecida como a etapa em que um sistema robótico colaborativo e um operador humano executam atividades simultaneamente, compartilhando o mesmo espaço de trabalho. Essa interação exige uma adaptação entre ambos os agentes para que suas habilidades complementares sejam integradas de maneira eficaz, permitindo a resolução de tarefas complexas (Baratta et al., 2023).

Segundo (Monostori et al., 2016; Nikolakis et al., 2019), o principal objetivo da implantação da robótica colaborativa na indústria é combinar as habilidades humanas com as capacidades robóticas, de forma a aumentar a produtividade, melhorar a flexibilidade e promover a ergonomia. Esse esforço é voltado para resolver tarefas complexas em um ambiente de trabalho dinâmico, observável e previsível. Frequentemente, o espaço compartilhado entre robôs colaborativos (*cobots*) e operadores humanos torna-se um espaço ciberfísico, unindo o mundo físico e o digital.

A implantação de *cobots* requer atenção especial à segurança, incluindo estratégias de mitigação de riscos nas fases de pré-colisão e pós-colisão (Haddadin, 2013). Além disso, a ergonomia é fundamental para garantir um ambiente favorável, especialmente em tarefas repetitivas, como operações de *pick and place*, montagem e transporte de cargas que, ao longo do tempo, podem causar danos à saúde do trabalhador (Cherubini et al., 2016; Pearce et al., 2018). A seguir, destacam-se algumas aplicações industriais dos *cobots* e seus respectivos benefícios:

1. **Montagem de Componentes** (Baratta et al., 2023)
    - **Aplicação:** Os *cobots* são utilizados na montagem de peças e componentes em linhas de produção, especialmente nas indústrias automobilísticas e eletrônicas.
    - **Benefícios:**
        - **Precisão e Consistência:** Garantem uniformidade na montagem, reduzindo erros humanos;
        - **Flexibilidade:** Podem ser reprogramados para diferentes tarefas, adaptando-se a mudanças no design dos produtos.
2. **Embalagem e Paletização** (Javaid et al., 2022)
    - **Aplicação:** Auxiliam no empacotamento de produtos e na organização de caixas em paletes para armazenamento ou envio.
    - **Benefícios:**

  - **Aumento da Produtividade:** Realizam tarefas repetitivas de forma contínua, acelerando o processo logístico;
  - **Redução de Lesões:** Minimizam o esforço físico dos trabalhadores, diminuindo o risco de lesões por esforço repetitivo.

3. **Inspeção de Qualidade** (Neves et al., 2024)

   - **Aplicação:** Utilizados para verificar a qualidade de produtos acabados, identificando defeitos ou inconsistências.
   - **Benefícios:**
     - **Detecção Precisa:** Capazes de identificar imperfeições minuciosas que podem passar despercebidas por inspeções humanas;
     - **Consistência:** Mantêm critérios de inspeção uniformes, assegurando padrões de qualidade elevados.

4. **Manuseio de Materiais Perigosos** (Pereira and Simonetto, 2021)

   - **Aplicação:** Empregados no manuseio de substâncias químicas ou materiais pesados que apresentam riscos à saúde humana.
   - **Benefícios:**
     - **Segurança:** Reduzem a exposição dos trabalhadores a ambientes ou materiais perigosos;
     - **Eficiência:** Realizam tarefas complexas com precisão, minimizando acidentes.

5. **Operações de Soldagem** (Fournier et al., 2023)

   - **Aplicação:** Desempenham atividades de soldagem em linhas de produção, especialmente na indústria metalúrgica.
   - **Benefícios:**
     - **Qualidade Consistente:** Asseguram soldas uniformes, melhorando a durabilidade dos produtos;
     - **Produtividade:** Operam de forma contínua, aumentando a capacidade produtiva.

A adoção de *cobots* nas indústrias não apenas otimiza processos produtivos, mas também promove um ambiente de trabalho mais seguro e ergonômico para os colaboradores humanos. Sua flexibilidade e capacidade de aprendizado permitem que se adaptem a diversas tarefas, tornando-se uma ferramenta valiosa na modernização industrial.

### 3.2.1 Robôs Colaborativos da Universal Robots

Segundo a Universal Robots (2021, 2024a), os robôs colaborativos da *Universal Robots* (UR) são robôs de peso ultraleve e estrutura compacta, projetados para trabalhar em conjunto com seres humanos em ambientes industriais compartilhados, sem a necessidade de barreiras de segurança tradicionais. Possuem

sensores inteligentes acoplados em sua estrutura, os quais permitem detectar forças contrárias e evitar colisões. Os *cobots* da UR são reconhecidos por sua flexibilidade, facilidade de uso e rápida implementação. Eles são empregados em uma variedade de aplicações, incluindo montagem, embalagem, paletização e inspeção de qualidade, contribuindo para o aumento da produtividade e eficiência operacional. Por ser compacto e dar a possibilidade de ser instalado diretamente dentro de máquinas ou em pequenos espaços de trabalho. A Figura 3.1 mostra a representação dos *cobots* da *Universal Robots*.

Figura 3.1: *Cobot* da Universal Robots (Universal Robots, 2024a)

Os robôs colaborativos *cobots* da linha *e-Series* da *Universal Robots*, que incluem os modelos UR3e, UR5e, UR10e, UR16e e UR20, apresentam pesos totais variando de $11kg$ a $64kg$ e capacidades de carga útil de $3kg$ a $20kg$, conforme as especificações de cada modelo. Todos os modelos possuem rotação de 360 graus em todas as juntas e rotação infinita na sexta junta (ou terceiro punho), proporcionando flexibilidade para diversas aplicações industriais, sua estrutura pode ser vista na Figura 3.2.

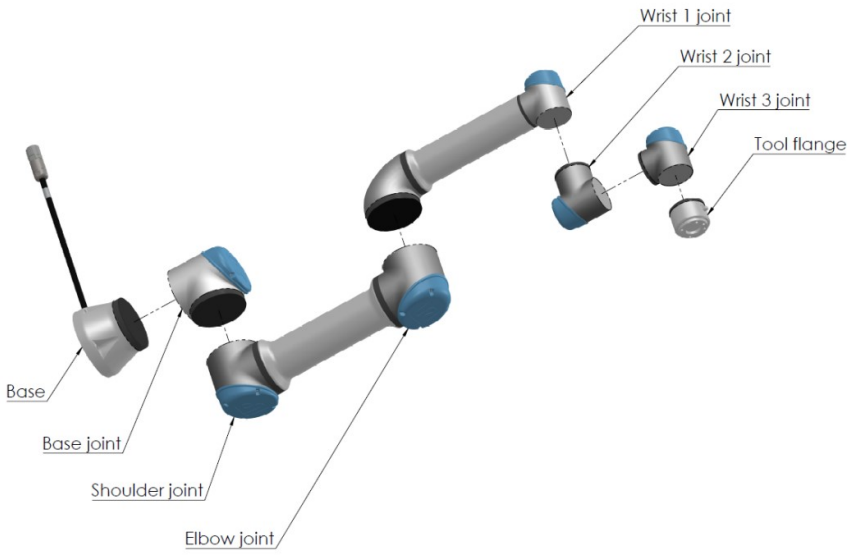



Figura 3.2: Sistema de Juntas (Universal Robots, 2021)

### 3.2.2 Sistema de Coordenadas e Orientação

Os robôs colaborativos da *Universal Robots*, por serem robôs da classe "manipulador", utilizam referenciais para definir a posição e a orientação de sua ferramenta no espaço de trabalho. Esses referenciais podem ser definidos em relação ao sistema de coordenadas cartesianas da base do robô ou ao centro da ferramenta (TCP) (Universal Robots, 2021).

A posição da ferramenta é representada em relação ao referencial da base por meio das coordenadas cartesianas $(x, y, z)$, que indicam a localização da origem da ferramenta no espaço. Já a orientação da ferramenta é especificada utilizando a representação vetor-ângulo, que descreve a rotação em torno de um eixo definido. Essa abordagem permite uma descrição precisa e intuitiva tanto da localização quanto da orientação da ferramenta em qualquer ponto do espaço de trabalho (Universal Robots, 2021; Meneguelli, 2022).

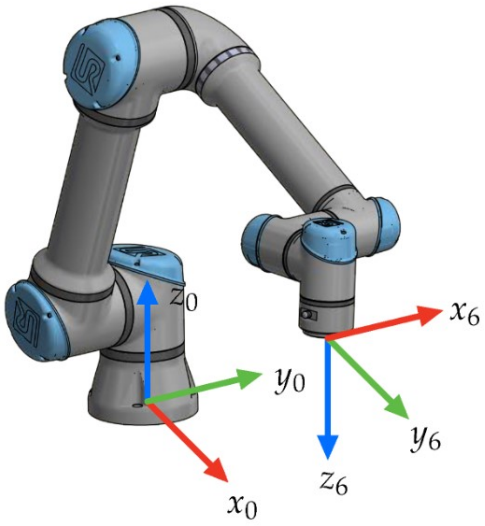



Figura 3.3: Coordenadas e Orientação do UR3.(Boschetti and Sinico, 2023)

Para esse *cobot*, define-se como eixo de referência a sua base, então a orientação da ferramenta é definida a partir de três parâmetros: $(r_x, r_y, r_z)$ ou seja, o vetor de rotação é calculado baseado nas componentes espaciais $(x, y, z)$ do vetor unitário $u$ e de sua rotação $\theta$ (Meneguelli, 2022; LaValle, 2006).

## 3.3 Programação de Robôs Colaborativos da Universal Robots

Os *cobots* da *Universal Robots* são amplamente conhecidos por sua flexibilidade e facilidade de uso, oferecendo diversas formas de programação para atender às variadas demandas industriais. Suas principais formas de programação são:

1. **Interface Gráfica Polyscope:** Uma interface de usuário intuitiva, baseada em tela sensível ao toque, que permite a programação dos robôs por meio da seleção de comandos pré-definidos. Esta abordagem elimina a necessidade de experiência prévia em programação e é ideal para criar programas simples e sequenciais de maneira rápida (Universal Robots, 2021);

2. **Programação Manual (*Free Drive*):** Permite mover fisicamente o braço robótico para ensinar posições e trajetórias. O movimento é registrado automaticamente no programa, facilitando a definição de trajetórias complexas. Esta abordagem é especialmente útil para usuários iniciantes e para tarefas que exigem ajustes frequentes (Universal Robots, 2023a);

3. **URScript:** Linguagem de programação proprietária da *Universal Robots*, que oferece controle avançado sobre o comportamento do robô. Com URScript, os programadores podem implementar soluções personalizadas, incluindo controle de força, lógica condicional e integração com outros dispositivos industriais (Universal Robots, 2023b);

4. **Integração com Sistemas CAD/CAM via Código G:**
Possibilita a utilização de trajetórias criadas em *softwares* CAD/CAM a partir de código G, amplamente utilizado em máquinas CNC. Esta abordagem é ideal para tarefas que requerem precisão em processos de manufatura, como usinagem e impressão 3D (Universal Robots, 2020);

5. **APIs e SDKs:** Os *cobots* oferecem suporte a APIs e SDKs que permitem a integração com sistemas externos, utilizando linguagens de programação como Python e $C++$. Isto possibilita o desenvolvimento de aplicações personalizadas e a conexão com sistemas de controle mais complexos (Universal Robots, 2023c; SDU Robotics, 2024);

6. **Programação via Python com UR-RTDE:** A biblioteca UR-RTDE (*Real-Time Data Exchange*) permite programar os *cobots* da UR utilizando Python. Com essa biblioteca, é possível enviar comandos de movimento, monitorar estados do robô em tempo real e implementar aplicações avançadas, como controle adaptativo e integração com sensores externos. EsTa abordagem é ideal para desenvolvedores que necessitam de flexibilidade e integração em sistemas de automação modernos (SDU Robotics, 2024);

7. **Uso de URCaps:** São módulos de *software* que estendem as funcionalidades do robô, permitindo integração com dispositivos e *software* de terceiros. Os URCaps são projetados para atender a necessidades específicas de aplicações industriais, oferecendo maior flexibilidade e escalabilidade (Universal Robots, 2023d).

### 3.3.1 Vantagens da Programação Flexível

Segundo Universal Robots (2024c), as diversas opções de programação tornam os *cobots* da *Universal Robots* acessíveis tanto para pequenas empresas quanto para grandes indústrias. Esta flexibilidade resulta em:

- **Facilidade de implementação:** Reduz o tempo necessário para configurar o robô em novos ambientes, além de ser possível modularizar o código para uma fácil replicação;

- **Adaptação a diferentes tarefas:** Permite reprogramar os *cobots* para executar diversas funções e *loop* de funções, desde montagem até inspeção de qualidade;

- **Integração com sistemas existentes:** As APIs e SDKs para *cobots* da UR permitem a integração com sistemas de controle, sensores externos e softwares CAD/CAM, ampliando as possibilidades de aplicação SDU Robotics (2024);
- **Redução do tempo de inatividade:** A reprogramação simples e rápida dos *cobots* facilita a troca de tarefas na linha de produção, minimizando períodos ociosos;
- **Escalabilidade:** Os *cobots* podem ser adaptados para atender a demandas crescentes de produção ou integrar novas tecnologias, como sensores inteligentes e sistemas IoT;
- **Flexibilidade em ambientes dinâmicos:** Podem ser facilmente realocados para diferentes áreas da fábrica, adequando-se a mudanças frequentes no *layout* ou nos processos produtivos.

## 3.4 Conclusão

Este capítulo abordou os fundamentos da robótica colaborativa e *cobots* da *Universal Robots* destacando os avanços tecnológicos, benefícios, aplicações industriais e programação. Foi enfatizada a importância dos robôs colaborativos no aumento da produtividade e na criação de ambientes de trabalho mais seguros e ergonômicos. Com o avanço da Quarta Revolução Industrial, os *cobots* se consolidam como ferramentas indispensáveis para enfrentar os desafios da modernização industrial, integrando eficiência, segurança e flexibilidade. No próximo capítulo será abordada a Metodologia deste trabalho, unindo fundamentos de visão computacional e robótica colaborativa para resolver um problema prático da indústria.

# Capítulo 4

# Metodologia

Esta seção descreve os procedimentos metodológicos adotados para o desenvolvimento deste trabalho, focado na automatização da inspeção visual de placas de circuito impresso (PCBs). Diante do problema proposto, foram exploradas e integradas abordagens avançadas de visão computacional e robótica colaborativa, resultando em um sistema capaz de identificar, segmentar e inspecionar as placas de forma eficiente. A solução desenvolvida utiliza visão computacional para segmentação e identificação de componentes nas PCBs, combinada com um robô colaborativo que automatiza a rotina de inspeção. Toda a implementação foi realizada na linguagem Python, garantindo flexibilidade e compatibilidade com ferramentas modernas de aprendizado de máquina e controle robótico. Este capítulo está estruturado em duas partes principais: *Hardware*, detalhando os componentes físicos como o robô colaborativo e garra, sistema de iluminação e suporte; e *Software*, abordando a movimentação e posicionamento orientado por visão computacional, além dos algoritmos de visão computacional desenvolvidos para a aplicação.

## 4.1 *Hardware*

Os *hardwares* selecionados para este projeto foram especificados com base em suas características técnicas, adequação às necessidades do sistema e facilidade de integração com os algoritmos de *software*. Os equipamentos foram destacados na Figura 4.1.

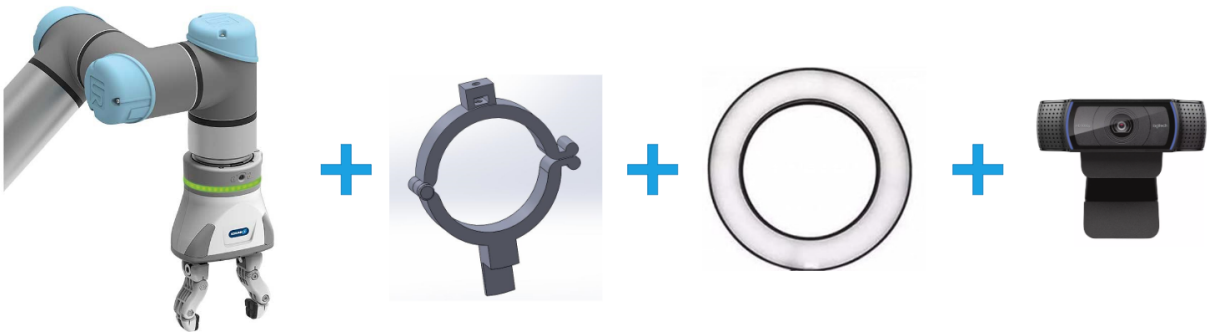


Figura 4.1: Equipamentos: UR5e, SCHUNK EGH, suporte para iluminação, *ring light* e Logitech C920e

O *cobot* UR5e foi escolhido por sua precisão, capacidade de carga útil de até 5 kg e compatibilidade com URCaps, que facilita a integração com a garra SCHUNK EGH. Além da segurança contra colisões e a não dependência de sistema de enclausuramento. Estas características permitiram a automação de tarefas com precisão, repetibilidade e segurança. No entanto, há limitações como a velocidade de movimentação e a rotação restrita a 360 em algumas juntas, consideradas durante o planejamento das tarefas.

A câmera Logitech C920e foi escolhida devido à sua resolução de 1080$p$, compatibilidade com bibliotecas de visão computacional, como OpenCV, e custo acessível. Ela atende à necessidade de capturar imagens 2D nítidas para processamento, embora dependa de iluminação adequada para um desempenho ideal. Para mitigar essa limitação, foi utilizado um *ring light*, que permite alternar entre luz amarela e branca durante os testes, oferecendo flexibilidade na iluminação. Além disso, foi desenvolvido um suporte de iluminação dimensionado com base nas medidas da garra, equipado com um regulador ajustável que possibilita adaptações a diferentes configurações. Esse suporte, confeccionado em uma impressora 3D, garantiu baixo custo de produção e alta flexibilidade.

Além disso, o suporte foi projetado para ser posicionado acima da câmera, evitando oclusão no campo de visão, e foi fixado com parafusos para maior estabilidade. Ele também inclui um apoio dedicado à iluminação, assegurando uniformidade e estabilidade da luz ao longo do trajeto do robô. Todo o sistema de câmera, iluminação e suporte foi desenvolvido para garantir uma iluminação uniforme e estável durante a operação, conforme ilustrado nas Figuras 4.1 e 4.2.

Por fim, escolheu-se a garra SCHUNK EGH por sua precisão e fácil integração com o UR5e. Sua capacidade de carga e design modular a tornam adequada para tarefas de manipulação de pequenos objetos. Contudo, sua limitação de carga (até 3 kg) restringe o uso para aplicações com objetos leves.

Vale ressaltar que como objeto de inspeção, foram escolhidas placas de Arduino, devido à sua ampla disponibilidade, baixo custo e características físicas adequadas, como tamanho compacto e componentes visíveis, em sua grande maioria, visto que possuem micro-componentes. Essas amostras permitiram a simulação de cenários reais de inspeção de placas eletrônicas, tornando-as ideais para testes práticos e validação do sistema desenvolvido.

O sistema composto por esses equipamentos e em ambiente de testes em laboratório podem ser vistos na Figura 4.2:

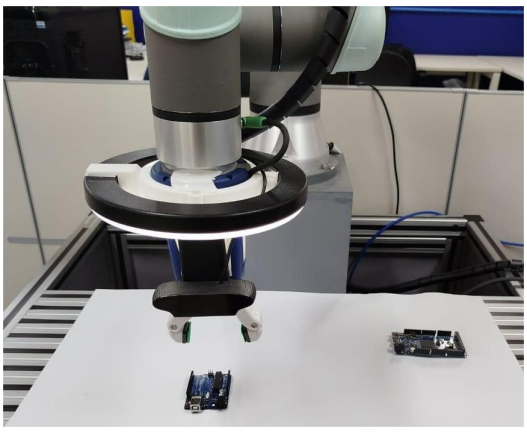


Figura 4.2: Equipamentos montados no robô em ambiente de trabalho

A montagem e integração desses equipamentos no robô segue os procedimentos de acordo com o Fluxograma 4.3.

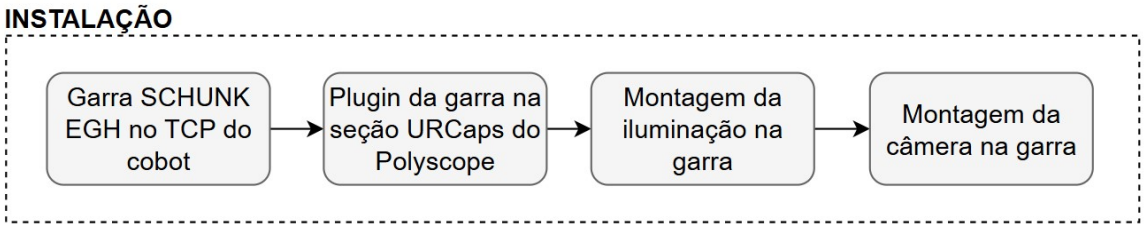



Figura 4.3: Fluxograma de instalação dos equipamentos

Nesse fluxograma, descreve-se o processo sequencial de montagem e configuração dos equipamentos utilizados no *cobot*: garra SCHUNK EGH, iluminação com suporte e câmera. O processo inicia-se com a fixação da garra SCHUNK EGH no TCP (*Tool Center Point*) do braço robótico. Em seguida, instala-se e condigura-se o *plugin* da garra no *software* PolyScope, na seção URCaps, por meio do *Teach Pendant*, possibilitando a integração entre o robô e a garra, além do controle preciso de suas funções. Na última etapa, a câmera é instalada na estrutura da garra, garantindo alinhamento com o centro da garra e considerando o deslocamento que há por estar acoplada externamente. Essas configurações foram projetadas para tornar possível a transformação do que a câmera vê para o que o robô irá pegar.

## 4.2 *Software*

Escolheu-se a linguagem Python para desenvolver o *software* do *cobot*, da visão computacional e da integração entre eles, por possuir todas as bibliotecas que satisfazem as necessidades desta pesquisa e ser muito utilizada em IA.

Para o funcionamento do *software* da visão computacional integrada ao *cobot*, é necessário instalar o seguinte conjunto de bibliotecas de acordo com a Tabela 4.1.

Tabela 4.1: Requisitos de *software*

| **Biblioteca** | **Versão** | **Biblioteca** | **Versão** |
|---|---|---|---|
| python | 3.10.5 | ur-rtde | 1.5.5 |
| opencv-python | 4.9.0 | numpy | 1.26.4 |
| matplotlib | 3.2.2 | pillow | 7.1.2 |
| pyYAML | 5.3.1 | requests | 2.23.0 |
| scipy | 1.4.1 | torch | 1.7.0 |
| torchvision | 0.8.1 | tqdm | 4.64.0 |
| pandas | 1.1.4 | seaborn | 0.11.0 |
| gradio | 3.35.2 | | |

Vale ressaltar que é necessário que o robô e o computador estejam na mesma rede.

### 4.2.1 *Software* de Movimentação do *cobot* UR

Foi desenvolvido um algoritmo em Python utilizando a API RTDE (*Real-Time Data Exchange*) para comandar o robô colaborativo em tempo real. A comunicação entre o robô e a API ocorre via interface IP. As funções principais da API para comandos do *cobot* estão listadas na Tabela 4.2, conforme SDU Robotics (2024).

Tabela 4.2: Principais Funções e Comandos da Biblioteca UR-RTDE (SDU Robotics, 2024)

| **Função/Comando** | **Descrição** |
|---|---|
| `RTDEControlInterface::moveL()` | Realiza movimento linear para uma posição alvo. |
| `RTDEControlInterface::moveJ()` | Realiza movimento articular para uma posição alvo. |
| `RTDEControlInterface::servoJ()` | Controla juntas em tempo real. |
| `RTDEControlInterface::speedJ()` | Define velocidades das juntas do robô. |
| `RTDEControlInterface::forceMode()` | Ativa modo de força para interação sensível. |
| `RTDEControlInterface::zeroFtSensor()` | Zera o sensor de força/torque do robô. |
| `RTDEControlInterface::setTcp()` | Define posição e orientação da ferramenta (TCP). |
| `RTDEControlInterface::setPayload()` | Ajusta a carga útil do robô. |
| `RTDEReceiveInterface::getActualQ()` | Obtém posições atuais das juntas. |
| `RTDEReceiveInterface::getActualTCPPose()` | Recupera posição e orientação atuais do TCP. |
| `RTDEReceiveInterface::getActualTCPSpeed()` | Obtém velocidade atual do TCP. |
| `RTDEReceiveInterface::getActualCurrent()` | Obtém correntes atuais das juntas. |
| `RTDEIOInterface::setStandardDigitalOut()` | Define estado das saídas digitais. |
| `RTDEIOInterface::setAnalogOutputVoltage()` | Ajusta tensão das saídas analógicas. |
| `RTDEIOInterface::setInputIntRegister()` | Define valor de registradores inteiros de entrada. |
| `RTDEIOInterface::setInputDoubleRegister()` | Define valor de registradores flutuantes de entrada. |
| `RTDEIOInterface::setToolDigitalOut()` | Controla o estado da saída digital da ferramenta. |

A função `RTDEControlInterface` é responsável pelo envio de comandos para controlar o *cobot* em tempo real, necessários para a executação dos movimentos lineares e articulares, pode-se através dela configurar parâmetros como a carga útil e o TCP, e ativar modos específicos, como o de força. Enquanto a função `RTDEReceiveInterface` é usada para monitorar o estado atual do *cobot*, fornecendo informações como a posição do TCP, ângulos das juntas, velocidades e leituras de sensores em tempo real. Já a função `RTDEIOInterface` é responsável pelas I/O do *cobot*, controlando os estados das entradas e saídas, assim possibilitando a manipulação dos sinais digitais e analógicos, além de configurar registradores, sendo utilizada para a integração de dispositivos externos, como garras e sensores. Com base nessas funções, os comandos mais utilizados estão listados na Tabela 4.3, a qual mostra o referencial de cada função, quesito essencial para que seja possível obter o movimento conforme esperado.

Tabela 4.3: Comandos de controle e monitoramento usados no *cobot* (SDU Robotics, 2024)

| **Função** | **Dados Retornados / Ação** | **Referencial** |
|---|---|---|
| `getActualTCPPose()` | Posição e orientação do TCP $(x, y, z, rx, ry, rz)$. | Base do robô (cartesiano). |
| `getActualQ()` | Ângulos das juntas $(q_1, q_2, q_3, q_4, q_5, q_6)$. | Articulação do robô. |
| `moveL()` | Move o TCP em linha reta. | Base do robô (cartesiano) |
| `moveJ()` | Move o TCP por trajetórias articulares. | Articulação do robô. |
| `setToolDigitalOut()` | Define estado da saída digital da garra. | Ferramenta (TCP). |

O comando `getActualTCPPose()` foi utilizado para localizar o robô no plano cartesiano, além de ter sido aplicado na validação dos pontos de parada do *cobot* ao longo da trajetória. Enquanto o comando `getActualQ()` foi utilizado para uma finalidade similar, porém voltada à orientação do *cobot*, visto que este comando provê os ângulos das juntas do robô, então ele foi aplicado na validação da rotação em $Rz$. Já o comando `moveL()`, foi usado para enviar a diretriz de trajetória linear para o *cobot*, para garantir que ele seguisse exatamente o percurso determinado. Enquanto o comando `moveJ()` atuou de forma parecida, porém relacionado ao ângulo de orientação $Rz$. E o comando `setToolDigitalOut()` foi aplicado na ação da garra, abertura e fechamento, funcionando como uma saída digital, utilizou-se este comando para chavear o estado da garra no momento do *Pick-and-Place*, conforme Algoritmo 1.

**Algorithm 1:** Lógica de controle da garra SCHUNK

**Input:** `action`: "close"ou "open"

```
if action == "close" then
    setToolDigitalOut(0, False);
    setToolDigitalOut(1, True);
else if action == "open" then
    setToolDigitalOut(0, True);
    setToolDigitalOut(1, False);
```

Vale ressaltar que as coordenadas do ponto-alvo da trajetória do *cobot* são determinadas pela visão computacional (item a ser explicado em seguida). Essas coordenadas incluem a posição no espaço cartesiano $(x, y, z)$ e a orientação $(rx, ry, rz)$, que são essenciais para guiar o robô até o local desejado. Nessa etapa, o *cobot* calcula a trajetória mais eficiente para atingir o ponto de destino, considerando o tipo de movimento e garantindo precisão e segurança.

### 4.2.2 *Software* de Visão Computacional: Técnicas Clássicas

Os algoritmos de implementados nessa etapa foram divididos em dois principais componentes: o processamento de imagens para a extração de informações relevantes e o controle do robô colaborativo para execução precisa das tarefas de alinhamento e manipulação, ou seja, o robô é guiado pela visão desde o momento que identifica a placa eletrônica até o momento que se posiciona da melhor forma para

executar o *Pick-and-Place*. Ambos os componentes trabalham de maneira integrada, conforme detalhado a seguir.

#### 4.2.2.1 Processamento de Imagens

O módulo de processamento de imagens foi desenvolvido utilizando a biblioteca OpenCV. Neste módulo foi possível programar funções para lidar com tarefas de visão computacional de forma modular e eficiente, permitindo o pré-processamento e a análise das imagens capturadas pela câmera, de acordo com a Tabela 4.4.

Tabela 4.4: Características principais da câmera (webcam Logitech C920e) (Logitech, 2024)

| **Característica** | **Descrição** |
|---|---|
| **Resolução Máxima** | 1920x1080p (Full HD) a 30 qps |
| **Campo de Visão** | 78° |
| **Foco Automático** | Ajuste automático da lente para objetos em diferentes distâncias |
| **Correção de Luz** | Ajuste automático para otimização em condições de baixa luz |

As principais etapas deste componente incluem:

1. **Captura de Imagem:** As imagens são capturadas pela câmera configurada para a resolução de $1920x1080$ pixels, utilizando alta qualidade para a análise da imagem;
2. **Pré-Processamento:**
   - Conversão para escala de cinza (*grayscale*) para simplificar o processamento da imagem em apenas um canal de cor;
   - Aplicação de limiarização binária inversa (*thresholding*) para destacar os objetos de interesse.
3. **Extração de Contornos:** Os contornos são identificados com base na imagem binarizada, utilizando algoritmos para filtragem de contornos por área e perímetro. Então o maior contorno identificado é selecionado como a região de interesse;
4. **Cálculo de Centro:** O centro geométrico do maior contorno é calculado e utilizado como referência para o alinhamento do robô;
5. **Cálculo da Rotação:** A rotação do objeto detectado é calculada e utilizada para o alinhamento da orientação do sexto eixo do robô, onde está acoplado a garra.

A integração desses métodos é realizada pelo módulo de localização de centro, que retorna as coordenadas centrais do objeto detectado, prontas para serem utilizadas pelo sistema de controle do robô para se alinhar em relação à placa, que será detalhado posteriormente.

#### 4.2.2.2 Controle do Robô

Como introduzido anteriormente, o controle do robô colaborativo foi implementado utilizando a API RTDE (*Real-Time Data Exchange Interface*) em Python. Então o sistema foi configurado para ajustar dinamicamente a posição do cobot com base nas informações obtidas pela câmera, para atribuir uma maior flexibilidade ao inspecionar produtos que possam estar em qualquer posição na linha de produção. As etapas principais incluem:

1. **Configuração Inicial:**
   - O robô é configurado com os parâmetros iniciais de posição relativa de alinhamento, $(x_h, y_h)$, além de uma tolerância de erro de posicionamento;
   - O comando `getActualTCPPose()` é utilizado para obter a posição inicial do robô no espaço de trabalho.
2. **Iteração de Correção:**
   - Os erros entre a posição do centro detectado pela câmera e o ponto-alvo do robô são calculados;
   - Pequenos deslocamentos (`dx`, `dy`) são estimados com base em um fator de ajuste (taxa de erro) e aplicados à posição atual do robô até que ele atinja a posição ideal;
   - O robô é movimentado para a nova posição usando o comando `moveL()`.
3. **Verificação da Tolerância:** A cada iteração, é verificado se o erro está dentro da tolerância especificada. Quando a tolerância é atingida, o robô para o movimento, indicando alinhamento com a placa.

Vale ressaltar que os parâmetros $x_h$ e $y_h$ representam a posição relativa do ponto central da placa, alinhada com a garra do robô, sendo esta a configuração ideal para a pega do robô. Como a câmera foi acoplada à garra do robô, o que cria um deslocamento fixo entre o centro da câmera e o centro da garra. Essa diferença foi considerada durante a calibração inicial para determinar os valores $x_h = 953$ e $y_h = 843$. Esses valores garantem que, ao alinhar o objeto detectado com as coordenadas $x_h$ e $y_h$, ele estará corretamente posicionado no centro da garra do robô para a manipulação.

O erro em $x$ e $y$ é normalizado pela largura e altura da imagem ($1920x1080$) capturada. A tolerância é verificada diretamente sobre esses valores normalizados. Por exemplo, se a tolerância for definida como 0.01, o sistema considera que o objeto está alinhado quando sua posição está dentro de 1% da largura e altura da câmera em relação ao ponto de referência $(x_h, y_h)$. Isso garante que o alinhamento seja avaliado de forma proporcional ao campo de visão da câmera.

Vale enfatizar também que a tolerância de erro define um limite aceitável de desvio entre a posição atual do objeto detectado e a posição-alvo $(x_h, y_h)$. Em termos práticos, essa tolerância é usada para determinar quando o sistema deve encerrar o movimento do robô. Todos os parâmetros iniciais de configuração estão na Tabela 4.5.

Tabela 4.5: Parâmetros iniciais de configuração

| **Parâmetro** | **Valor** |
|---|---|
| $x_h$ (`home_x`) | 953 pixels |
| $y_h$ (`home_y`) | 843 pixels |
| Fator | 0.3 |
| Tolerância de deslocamento | 0.01 |
| Tolerância do ângulo | 5 graus |
| Altura entre a base e a garra | 10 cm |
| Altura entre a base e a câmera | 17 cm |

### 4.2.3 Integração dos Componentes

O sistema integra o processamento de imagens e o controle do robô em um laço iterativo. Inicialmente, a câmera captura uma imagem que é processada para localizar o centro do objeto-alvo. Em seguida, o robô é movimentado de maneira incremental para corrigir os erros detectados, até que a posição final esteja dentro da tolerância estabelecida.

A arquitetura do sistema pode ser descrita pelos seguintes passos:

1. Capturar a imagem;
2. Processar a imagem para identificar o objeto e o seu centro;
3. Calcular os erros de posicionamento;
4. Movimentar o robô para corrigir os erros;
5. Repetir o processo até que o erro seja menor que a tolerância;
6. Calcular a rotação do objeto com posicionamento alinhado;
7. Alinhar a orientação da garra;
8. Inspecionar se a placa está conforme item cadastrado;
9. Realizar o *Pick-and-Place* de acordo com sua classificação (conforme ou não conforme).

De acordo com essa arquitetura, montou-se o Algoritmo 2 referente ao fluxo geral de funcionamento do sistema com a integração do algoritmo de visão computacional com o do robô para que seja possível guiá-lo por meio da câmera. Vale ressaltar a inclusão do módulo de inspeção da qualidade e do módulo de manipulação e separação por meio da classificação.

**Algorithm 2:** Fluxo Geral do Robô Guiado por Visão com Inspeção de Qualidade

**Input:** `home_x`, `home_y`, `fator`, tolerâncias (`tol_x`, `tol_y`, `tol_rz`), `objeto_altura`, `destino_OK`, `destino_NG`

**Result:** Peça analisada, manipulada e classificada (OK ou NG)

**Etapa 1: Inicialização**
Configurar câmera, parâmetros do robô (`home_x`, `home_y`, `fator`, tolerâncias), e estados iniciais;
**Etapa 2: Localização e Alinhamento**
**while** *Erro de posição (`error_x`, `error_y`) > tolerância* **do**
    Capturar imagem e processar posição do objeto (`center_x`, `center_y`);
    Calcular `error_x`, `error_y`;
    Atualizar `pose.x`, `pose.y` e enviar comando para ajustar;
**while** *Erro angular (`error_rz`) > tolerância* **do**
    Capturar imagem e processar ângulo do objeto (`angle`);
    Calcular `error_rz`;
    Atualizar junta $Rz$ com incremento controlado e enviar comando;
**Etapa 3: Inspeção de Qualidade**
Capturar nova imagem para análise de qualidade;
**if** *Objeto está no padrão (`Qualidade = OK`)* **then**
    Definir `destino` como `destino_OK`;
**else**
    Definir `destino` como `destino_NG`;
**Etapa 4: Manipulação e Classificação**
Realizar Pick-and-Place para `destino`;
Liberar peça (`gripper_release`) e retornar à posição inicial;

### 4.2.4 Cálculos do Sistema de Robótica e Visão

O sistema utiliza uma série de cálculos para alinhar o robô ao objeto detectado pela visão computacional para a realização do *Pick-and-Place*. As etapas principais incluem cálculo de erro, deslocamentos necessários, atualização da posição do robô, e critérios de parada.

#### 4.2.4.1 Cálculo do Erro

O cálculo do erro é fundamental para determinar o quanto a posição atual do robô está desalinhada em relação ao ponto-alvo $(x_h, y_h)$ de alinhamento. As divisões pela altura e largura da imagem são usadas para normalizar os erros com base nas dimensões da imagem, como citado anteriormente. A magnitude do erro em cada eixo é obtida pelas equações (4.1) e (4.2).

$$\epsilon_x = \frac{h_x - c_x}{l_{image}} \tag{4.1}$$

$$\epsilon_y = \frac{h_y - c_y}{h_{image}} \tag{4.2}$$

A magnitude do erro ($\epsilon$) é então calculada utilizando a norma Euclidiana, conforme a equação (4.3):

$$\epsilon = \sqrt{\epsilon_x^2 + \epsilon_y^2} \tag{4.3}$$

Caso a magnitude do erro seja menor ou igual à tolerância do erro, considera-se que o robô está alinhado.

#### 4.2.4.2 Cálculo dos Deslocamentos

Os deslocamentos necessários para corrigir o erro nos eixos $x$ e $y$ são calculados com base no fator de ajuste (`fator`), de acordo com as equações (4.4) e (4.5). Ajustados pelos divisores 5 e 9 para controlar a sensibilidade da velocidade do deslocamento.

$$\mathrm{dx} = \frac{\eta \cdot \epsilon_x}{5} \tag{4.4}$$

$$\mathrm{dy} = \frac{-\eta \cdot \epsilon_y}{9} \tag{4.5}$$

#### 4.2.4.3 Atualização da Posição do Robô

A cada iteração, a posição do robô é ajustada somando-se os deslocamentos (`dx`, `dy`) às coordenadas atuais nos eixos $x$ e $y$, equações (4.6) e (4.7) respectivamente.

$$p'_x = p_x + dx \tag{4.6}$$

$$p'_y = p_y + dy \tag{4.7}$$

onde, $p'_x$ é a posição atual no eixo $x$ somado ao deslocamento, assim como para $p'_y$ para o eixo $y$.

#### 4.2.4.4 Cálculo do Ângulo de Orientação

O cálculo do ângulo é realizado para garantir que o robô esteja corretamente alinhado com o objeto detectado. Esse ângulo é obtido através do cálculo do menor retângulo rotacionado que circunscreve o contorno do objeto. A orientação do objeto é determinada com base no ângulo do retângulo, ajustando-o para alinhar corretamente o lado maior do objeto. Caso o ângulo detectado não esteja conforme o valor esperado (90) graus, o robô ajusta o terceiro pulso ($Rz$) para compensar o desvio e garantir o alinhamento adequado. Com base nisto, o robô sabe se deve rotacionar ($Rz$) para a direita ou para a esquerda.

- **Cálculo do retângulo mínimo rotacionado:** O retângulo mínimo rotacionado é definido como o menor retângulo que pode ser rotacionado para circunscrever completamente os pontos de um

contorno detectado em uma imagem. O processo para a obtenção do retângulo mínimo rotacionado é obtido da seguinte maneira:

1. **Entrada (Contorno):** O contorno do objeto é representado por um conjunto de $n$ pontos no plano bidimensional da imagem:

$$C = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \tag{4.8}$$

2. **Fecho Convexo:** O menor polígono convexo que circunscreve todos os pontos do contorno $C$ é calculado, conhecido como o fecho convexo:

$$H = (x'_1, y'_1), (x'_2, y'_2), \ldots, (x'_k, y'_k), \quad k \leq n \tag{4.9}$$

3. **Busca pelo Retângulo Rotacionado:** Entre todos os retângulos que podem ser rotacionados para circunscrever $H$, o retângulo com a menor área é selecionado:

$$\text{Área} = \min(\text{largura} \times \text{altura}) \tag{4.10}$$

   Para cada retângulo avaliado, as arestas do fecho convexo são utilizadas como possíveis orientações do retângulo.

4. **Resultado:** O retângulo rotacionado mínimo é descrito por:
   - O **centro** do retângulo $(cx, cy)$;
   - Suas **dimensões** $(largura, altura)$;
   - O **ângulo de rotação** $\theta$, que é o ângulo entre o lado maior do retângulo e o eixo horizontal.

A orientação do retângulo segue as seguintes convenções:

- $\theta \in [-90°, 0°]$: O lado maior está inclinado no sentido anti-horário em relação ao eixo $x$;
- Se o lado menor do retângulo é considerado a largura ($largura > altura$), o ângulo é ajustado adicionando 90° para garantir consistência.

- **Detecção do Ângulo do Objeto:** O ângulo detectado é representado por $(Rz_{cam})$, obtido através do retângulo mínimo que envolve o objeto, conforme a condição:

$$Rz_{cam} = \begin{cases} \text{Ângulo fornecido pelo retângulo mínimo}, & \text{se largura} > \text{altura} \\ \text{Ângulo fornecido pelo retângulo mínimo} + 90°, & \text{caso contrário.} \end{cases}$$

  Essa condição garante que o lado maior do retângulo, definido como largura, esteja sempre corretamente orientado em relação ao sistema de coordenadas do robô, conforme explicado anteriormente.

- **Erro Angular:** O erro angular $(\Delta Rz_{cam})$ é a diferença entre o ângulo esperado pela câmera $(Rz_{cam(ref)})$ e o ângulo atual detectado, calculado pela equação:

$$\Delta Rz_{cam} = \left(Rz_{cam(ref)} - Rz_{cam} + 180^\circ\right) \bmod 360^\circ - 180^\circ \tag{4.11}$$

- **Ângulo Desejado para o Robô:** O ângulo desejado para o terceiro pulso do robô ($Rz_{cobot}$) é calculado considerando o ângulo esperado do robô quando alinhado ($Rz_{cobot(ref)}$) e o erro angular detectado pela câmera:

$$Rz_{cobot} = Rz_{cobot(ref)} - \Delta Rz_{cam} \tag{4.12}$$

  Este valor é normalizado no intervalo $[-180^\circ, 180^\circ]$ pela equação:

$$Rz_{cobot} = (Rz_{cobot} + 180^\circ) \bmod 360^\circ - 180^\circ \tag{4.13}$$

- **Condição de Alinhamento:** O alinhamento angular é considerado concluído quando a diferença absoluta entre o ângulo atual do robô ($Rz'_{cobot}$) e o ângulo desejado ($Rz_{cobot(ref)}$) está dentro da tolerância de ângulo pré-definida:

$$Rz_{cobot} - Rz'_{cobot} < Rz_{tol} \tag{4.14}$$

#### 4.2.4.5 Cálculo da Altura para Manipulação do Objeto

Para o movimento no eixo $z$, calcula-se a altura necessária para que o *cobot* desça e pegue a placa, levando em consideração que neste momento eles já estão alinhados. E sabendo que no eixo $z$ há um *offset* pré-definido que depende da altura entre objeto e a ponta da garra do robô, de acordo com (4.15).

$$Z = Z_{cobot} - Z_{obj} \tag{4.15}$$

Após a manipulação, o *cobot* retorna à altura inicial adicionando o *offset* de subida, para este caso, foi adotado o mesmo *offset* tanto para descida, quanto para subida, equação (4.16).

$$Z = Z_{cobot} + Z_{obj} \tag{4.16}$$

Essa volta para a posição anterior foi adotada para esse ponto guiado pela visão ser também um ponto intermediário no *Pick-and-Place*, e facilitar o movimento durante a trajetória em que o *cobot* deixa a placa no devido lugar.

### 4.2.5 Sistema de Inspeção da Placa

A inspeção de qualidade dos produtos será divida em duas fases: cadastro e inspeção em tempo real para separação conforme categoria (OK) ou (NG), conforme e não conforme.

#### 4.2.5.1 Cadastro da Placa

O cadastro é realizado previamente e uma única vez para cada novo modelo de placa, ou seja, será cadastrado com somente uma foto do produto em seu melhor estado para ser usado como padrão. Não é necessário que o operador faça anotação de nenhum componente da placa, basta tirar uma foto que alimentará o algoritmo de segmentação de componentes. O fluxo automático do algoritmo utilizado para o cadastro está detalhado na Figura 4.4.

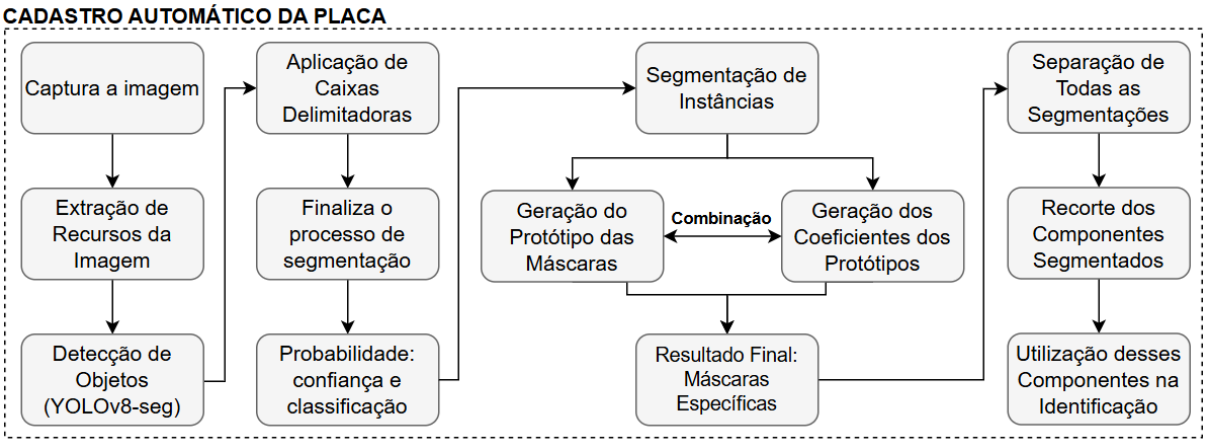



Figura 4.4: Segmentação por Instâncias do Cadastro da PCB usando FastSAM

Após a câmera realizar a captura da imagem da placa eletrônica, inicia-se o processo automático de segmentação utilizando o modelo FastSAM, no qual ocorre a segmentação de cada componente eletrônico da placa e a aplicação de uma caixa delimitadora chamada *bounding box* indicando cada segmentação.

O FastSAM é um modelo destinado à segmentação de todas as instâncias possíveis em um único tiro *One-Shot* nesse caso. Ao utilizar o modelo YOLOv8-seg como base, um avançado detector de objetos, permite identificar e segmentar todas as instâncias presentes na imagem de maneira eficiente, gerando máscaras únicas correspondentes a cada componente detectado. Junto disso, são geradas caixas delimitadoras ao redor do componente segmentado, que contém as coordenadas que definem a localização de cada objeto na imagem e as probabilidades associadas à presença de um objeto e sua respectiva classe. Durante a fase de segmentação de instâncias, há em paralelo o processo de geração do protótipo de máscaras de segmentação e a geração dos coeficientes (pesos) para esses protótipos de máscaras, que ao combiná-los ocorre o resultado das máscaras finais.

Após todos os componentes segmentados, tem-se a etapa de "recorte", denominado na literatura como *crop*, para gerar os *templates* individuais dos componentes que serão buscados nas placas seguintes ao longo do processo de inspeção, a partir da técnica de *Template Matching*.

#### 4.2.5.2 Identificação de Padrão

O Template Matching é uma técnica de visão computacional usada para localizar uma região específica, *template*, dentro de uma imagem maior, todo o campo de visão da câmera. Essa técnica funciona ao calcular a correlação entre o *template* e as sub-regiões da imagem, avaliando o quão bem cada sub-região

corresponde ao *template*. Partindo deste princípio, trabalhou-se no módulo de identificação do padrão da placa. A partir do cadastro da placa, obteve-se o padrão ideal, com isto, gerou-se automaticamente pequenos *templates* de cada componente da placa, os quais serão buscados na fase de identificação de padrão, por meio do Template Matching. O fluxo automático dessa busca encontra-se detalhado na Figura 4.5.

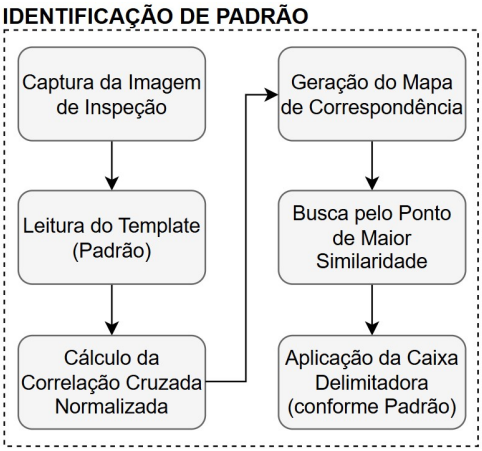



Figura 4.5: Busca pelo padrão na nova imagem de inspeção

De acordo com a Figura 4.5, tem-se que o *Template Matching* faz a verificação da presença e/ou ausência dos componentes na placa, utilizando uma abordagem baseada na **correlação cruzada normalizada** para medir a similaridade entre o template (imagem padrão) gerado pelo Cadastro da Placa, e as sub-regiões da imagem da placa que está sendo inspecionada. Durante o processo de verificação, o template dos componentes é deslizado sobre a imagem da placa, de forma semelhante a uma convolução 2D, gerando um mapa de correspondência que avalia o alinhamento do template com diferentes partes da imagem. Durante esse processo, é calculada a métrica de similaridade para cada posição na imagem, resultando em valores no intervalo $[-1, 1]$, onde valores mais próximos de 1 indicam correspondência perfeita.

A partir disso, é determinado o ponto de maior correspondência (valor máximo) no mapa, indicando a posição mais provável do template na placa. Essa informação é utilizada para verificar a presença e o alinhamento de componentes em sua forma padrão. Com base na taxa de correlação obtida, o algoritmo classifica a placa como **"conforme"** ou **"não conforme"**. Posteriormente, esta decisão é utilizada pelo *cobot* para a tomada de ação correspondente, como rejeitar placas com componentes ausentes ou desalinhados, garantindo a precisão e a qualidade do processo de inspeção.

Vale ressaltar que o registro da placa é realizado no início do processo e uma única vez, sem necessitar de anotação manual dos componentes, ou seja, é necessário apenas uma imagem da placa. Após isto, inicia-se o processo no qual o robô colaborativo realiza a atividade de *Pick-and-Place*: localizando a placa por meio da câmera; movendo-se até à placa, caso esta esteja deslocada ou rotacionada; e realizando a ação de separar as placas "conformes"das "não conformes", repetindo o fluxo até todas as placas serem inspecionadas, conforme detalhado na Figura 4.6.

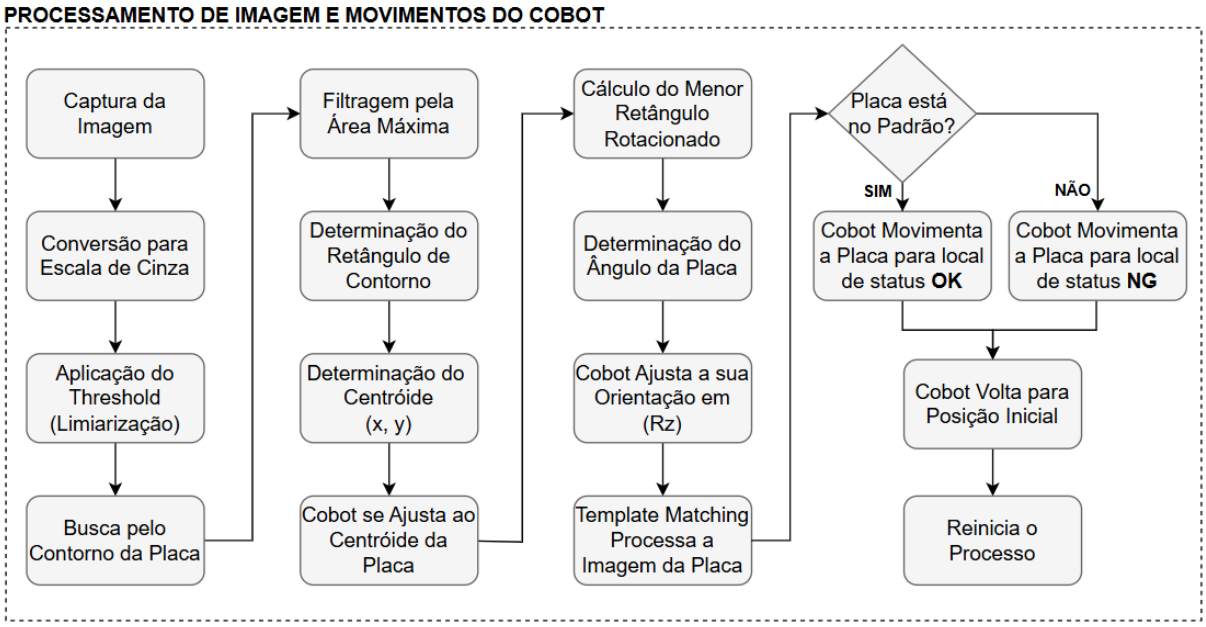



Figura 4.6: Fluxograma geral do processo de localização da placa, movimentação do cobot e determinação do estado da placa

### 4.2.6 Testes Práticos

Os testes praticos foram divididos em duas etapas: cadastro automático juntamente com a identificação do padrão da placa, e alinhamento nos eixos $(x, y)$ e orientação $R_z$ do *cobot*.

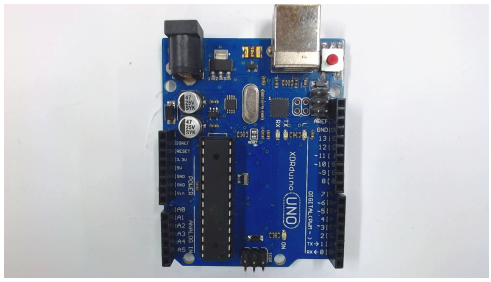
#### 4.2.6.1 Cadastro Automático da Placa

Para o cadastro do padrão ideal de placa, a ser utilizado como referência para as placas inspecionadas ao longo da linha de produção, foram propostos três cenários de testes com alturas de 11 cm, 15 cm, 17.5 cm e 20 cm. Esses valores foram definidos considerando o campo de visão da câmera que deve englobar toda a placa e as características básicas da imagem capturada de acordo com a Tabela 4.6.

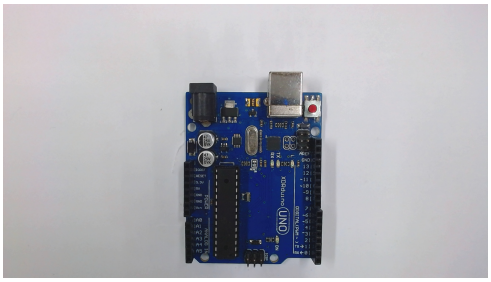
Tabela 4.6: Configurações da câmera e imagem

| **Parâmetro** | **Valor** |
|---|---|
| Altura (cenários) | 11 cm, 15 cm, 17.5 cm, 20cm |
| Brilho | 151 |
| Contraste | 122 |
| Saturação | 255 |
| Nitidez | 128 |

Para cada imagem deste teste, serão analisados: foco, uniformidade na iluminação e campo de visão.



(a) Distância 11cm, sem iluminação

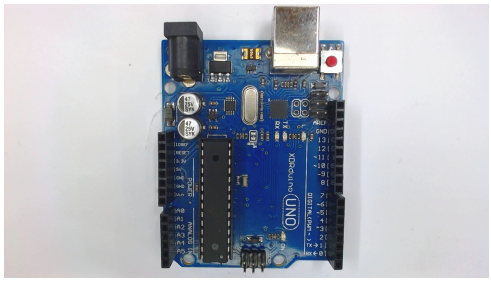


(b) Distância 15cm, sem iluminação

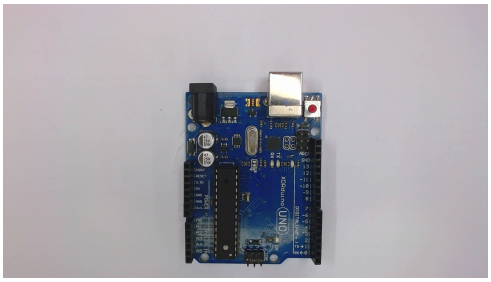
(c) Distância 17.5cm, sem iluminação

(d) Distância 20cm, sem iluminação

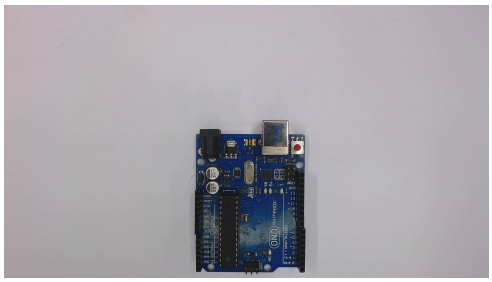
Figura 4.7: Teste de altura para cadastro automático (sem iluminação)



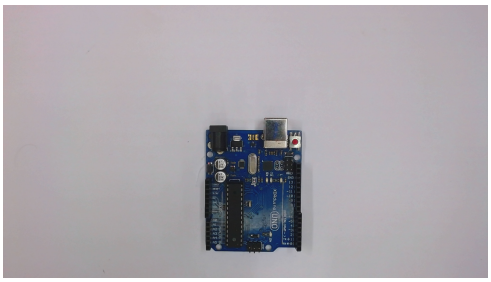
(a) Distância 11cm, com iluminação



(b) Distância 15cm, com iluminação



(c) Distância 17.5cm, com iluminação



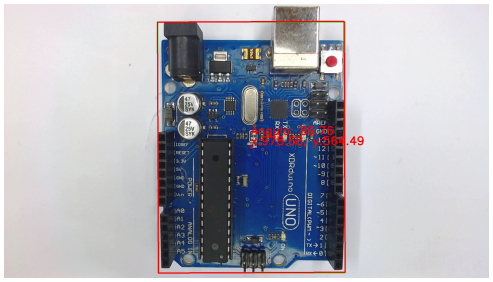
(d) Distância 20cm, com iluminação

Figura 4.8: Teste de altura para cadastro automático (com iluminação)

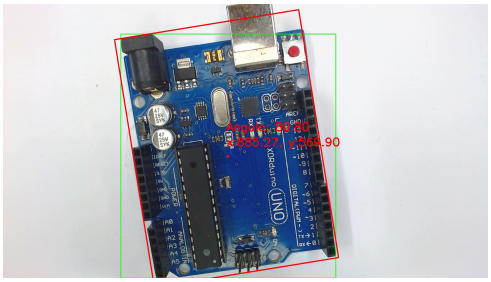
#### 4.2.6.2 Deslocamento do *cobot*

O segundo teste aborda a detecção da placa, o cálculo do $(x, y)$ de alinhamento entre *cobot* e placa eletrônica, assim como a sua orientação $Rz$. Para cada altura, foram analisados:

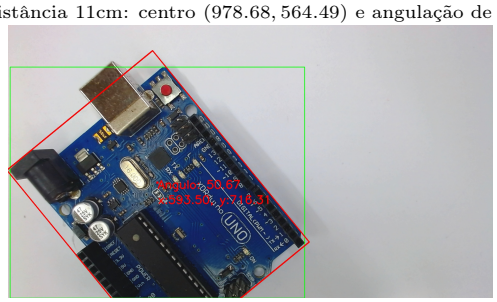
- Placa alinhada no centro e rotacionada:



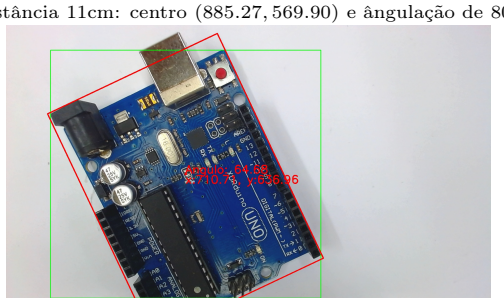
(a) Distância 11cm: centro $(978.68, 564.49)$ e angulação de $89.76°$



(b) Distância 11cm: centro $(885.27, 569.90)$ e ângulação de $80.80°$



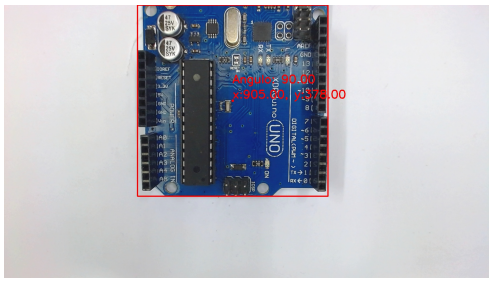
(c) Distância 11cm: centro $(593.50, 716.31)$ e angulação de $50.67°$



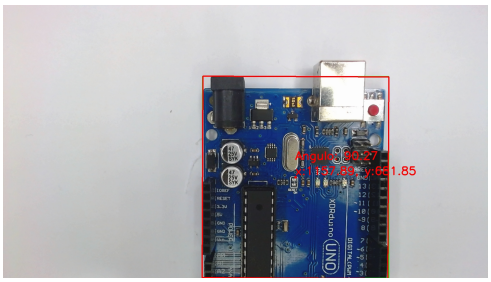
(d) Distância 11cm: centro $(710.71, 636.96)$ e angulação de $64.58°$

Figura 4.9: Teste de detecção da placa, centro e angulação

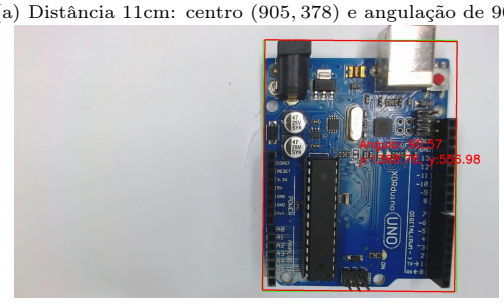
- Placa deslocada nos eixos $(x, y)$ e rotacionada:



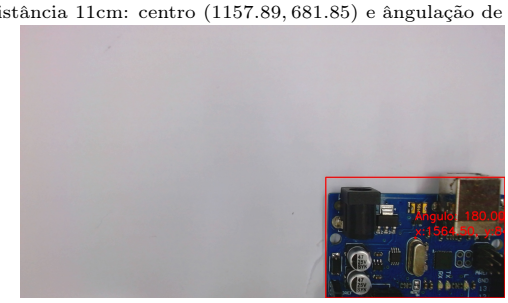
(a) Distância 11cm: centro $(905, 378)$ e angulação de $90°$



(b) Distância 11cm: centro $(1157.89, 681.85)$ e ângulação de $90.27°$



(c) Distância 11cm: centro $(1368.76, 556.98)$ e angulação de $90.57°$



(d) Distância 11cm: centro $(1568.50, 841)$ e angulação de $180°$

Figura 4.10: Teste de detecção da placa, centro e angulação com deslocamento

#### 4.2.6.3 Teste de Segmentação da Placa

O teste de segmentação das PCBs foi realizado utilizando os mesmos parâmetros no algoritmo para diferentes alturas da câmera em relação à base, com o objetivo de verificar o impacto da variação de distância na precisão da detecção e identificação dos componentes. As Figuras 4.11, 4.12, 4.13 e 4.14 apresentam a mesma placa segmentada nas distâncias de $11cm$, $15cm$, $17.5cm$ e $20cm$, respectivamente.

Essa avaliação busca entender como a alteração da altura da câmera pode impactar na segmentação dos

componentes, especialmente em cenários onde o campo de visão e a resolução aparente da placa variam. As diferentes alturas foram selecionadas para simular condições reais de uso, permitindo identificar possíveis limitações ou ajustes necessários no algoritmo para aplicações em distâncias específicas.

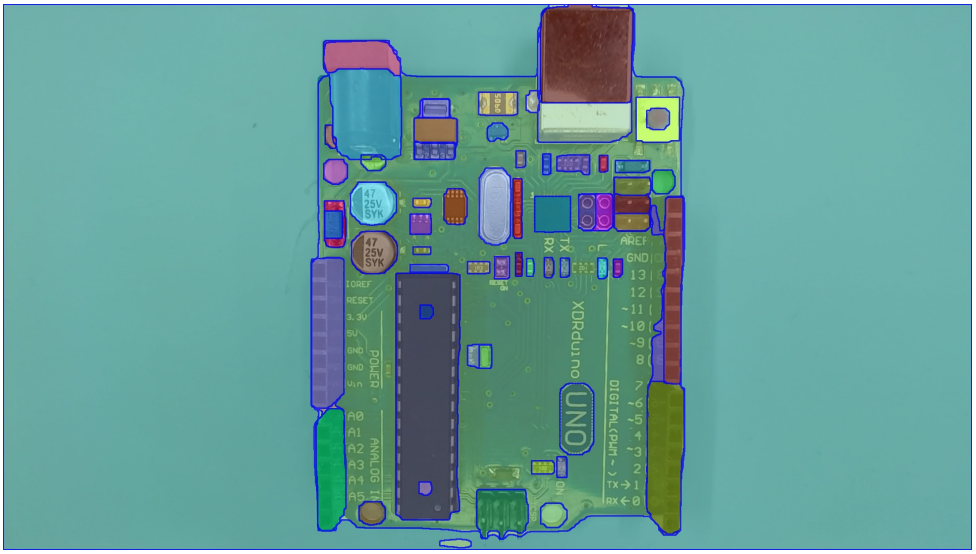



Figura 4.11: Segmentação aplicada à uma distância de 11cm

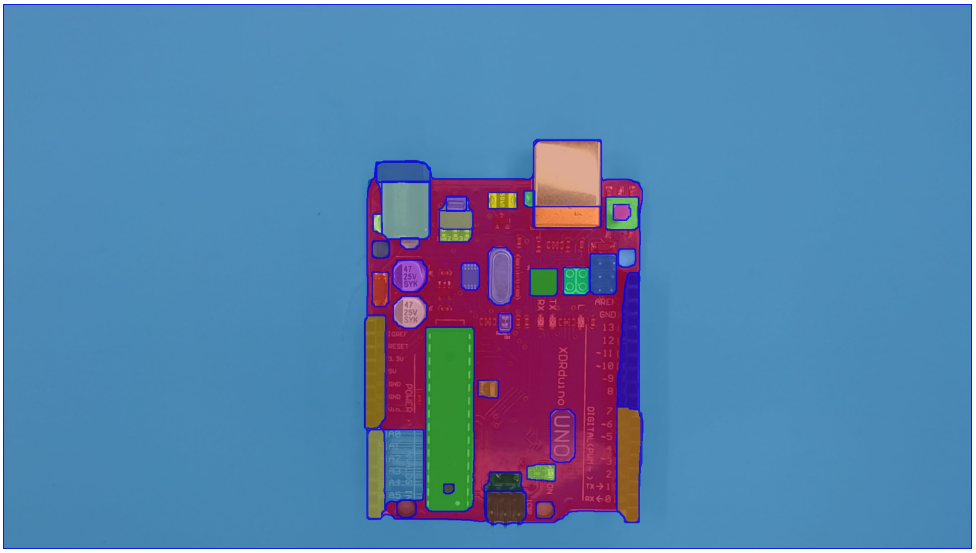



Figura 4.12: Segmentação aplicada à uma distância de 15cm

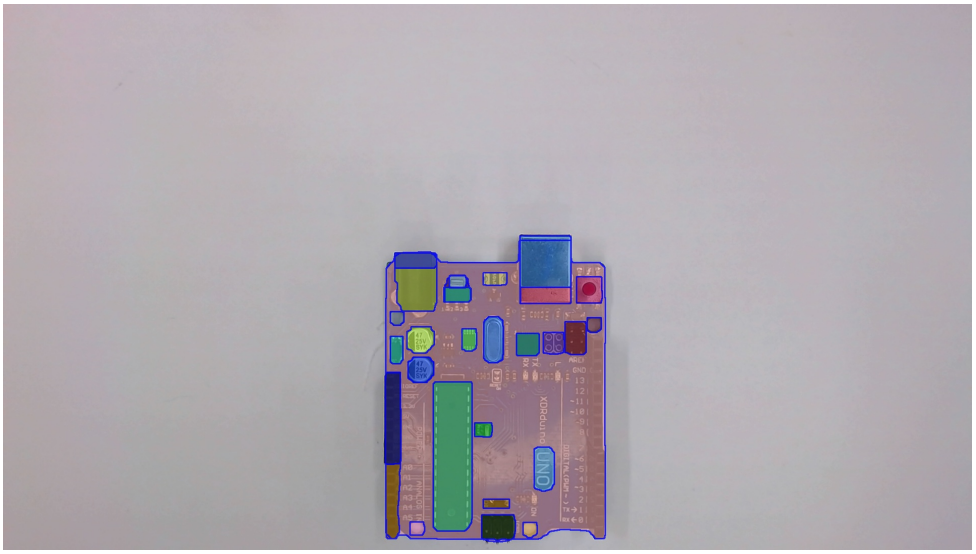


Figura 4.13: Segmentação aplicada à uma distância de 17.5cm

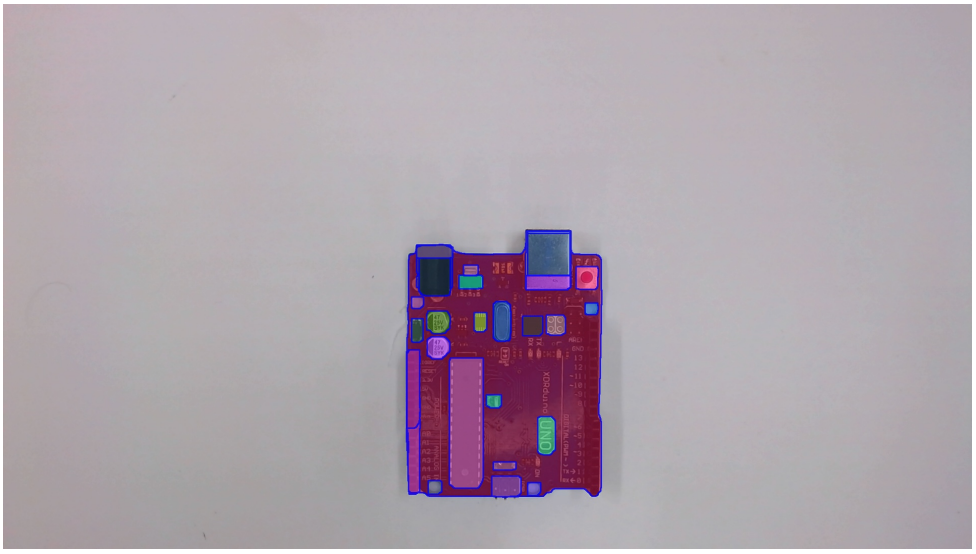


Figura 4.14: Segmentação aplicada à uma distância de 20cm

## 4.3 Conclusão

Este capítulo apresentou os procedimentos metodológicos de toda a arquitetura do sistema, abordando em cada seção o passo a passo necessário para a elaboração do projeto e detalhando os testes práticos realizados.

# Capítulo 5

# Resultados

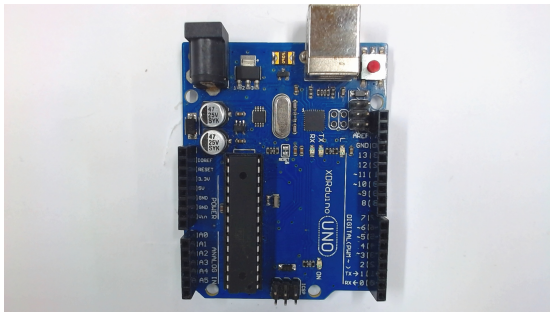
Esta seção visa apresentar os resultados obtidos à partir dos testes experimentais realizados em laboratório para cada parte da arquitetura do sistema abordada na seção de metodologia. Estes resultados serão divididos em:

1. Captura da Imagem;
2. Detecção da PCB na Imagem;
3. Cadastro Automático da PCB;
4. Identificação dos Componentes da PCB;
5. Métricas de Avaliação;
6. Movimentação do Robô Colaborativo.

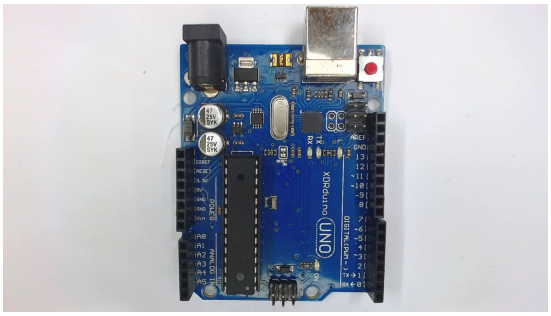
## 5.1 Captura da Imagem

Para o teste de captura de imagem, analisou-se a câmera *webcam* por meio de suas especificações. A utilização de *webcam* como câmera em visão computacional é de fácil implementação, apesar de suas limitações.

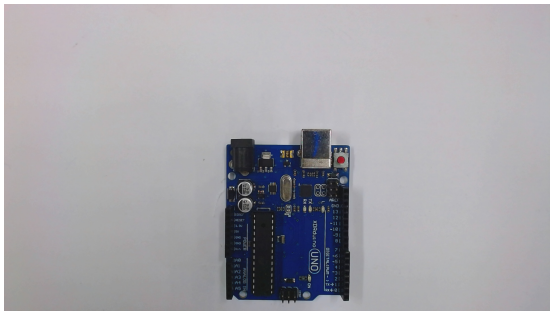
Foi implementado um algoritmo para captura da imagem, analisou-se as imagens em diferentes alturas e com iluminação e sem. Como a câmera é 2D, buscou-se uma altura adequada para ter uma boa relação entre campo de visão e placa (objeto) a ser analisado em diferentes posições e angulações. Alguns exemplos destes testes estão na Figura 5.1.



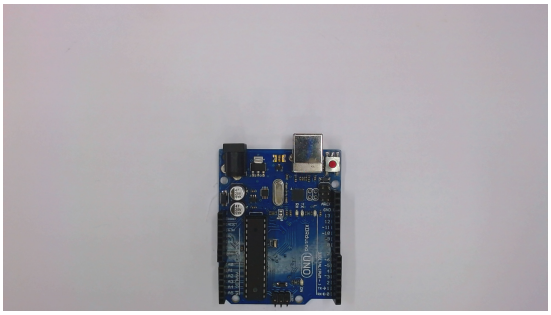
(a) Distância 11cm, sem iluminação



(b) Distância 11cm, com iluminação



(c) Distância 17.5cm, sem iluminação



(d) Distância 17.5cm, com iluminação

Figura 5.1: Captura de imagem para definição de altura e iluminação

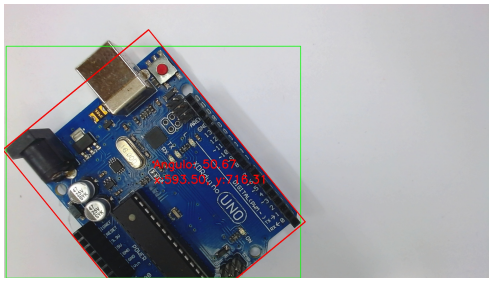
## 5.2 Detecção da PCB na Imagem

Diante dos testes de altura e iluminação, definiu-se a iluminação branca alta como padrão e distância da base do objeto para a câmera de 18cm. Esta configuração foi necessária para que a câmera pudesse capturar a imagem da placa em um campo de visão mais amplo.

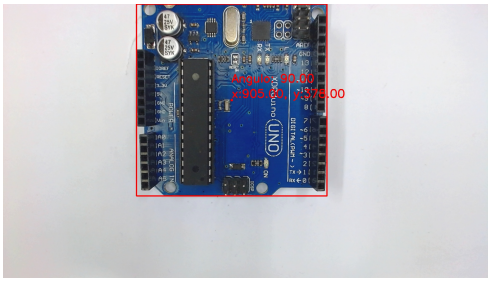
Com o campo de visão amplo, foram realizados diversos testes de posicionamento e rotação para por em prova o algoritmo de alinhamento de $x$, $y$ e $Rz$ do robô. Foi definida também o alinhamento padrão da garra e objeto, configuração a ser utilizada como pontos de referência.

No desenvolvimento do algoritmo de detecção de PCB, transformou-se a imagem de capturada para tons de cinza, a fim de diminuir o canal de cores e facilitar o processamento. Após isto, aplicou-se o *threshold* com limiar de 100 e do tipo *Inverse-Binary Thresholding*, o qual atribui ao pixel de destino o valor zero, se o pixel de origem correspondente for maior que o limite. Realizou-se uma filtragem por contorno, que busca contornos externos de objetos ao longo da imagem. Aplicou-se também um filtro para buscar a maior área, a fim de procurar o contorno externo da placa e detectá-la. Atribuiu-se a *Bounding Box* para estes objetos selecionados.

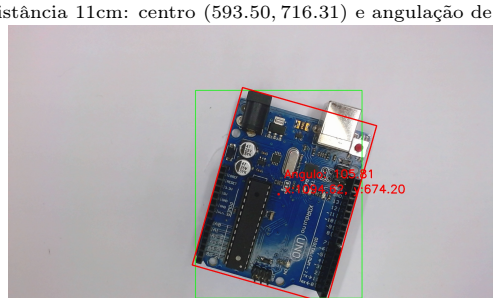
Para ficar evidente na imagem processada, gerou-se o desenho do retângulo indicando onde o objeto está, baseado em sua borda exterior e a partir destas dimensões, determinou-se o centróide do retângulo. Com isso, obteve-se o resultado mostrado na Figura 5.15.



(a) Distância 11cm: centro $(593.50, 716.31)$ e angulação de 50.67°



(b) Distância 11cm: centro $(905, 378)$ e angulação de 90°



(c) Distância 15cm: centro $(1094.62, 674.20)$ e angulação de 105.81°


(d) Distância 17.5cm: centro $(726.76, 188.37)$ e angulação de 178.66°

Figura 5.2: Validação do algoritmo de detecção de coordenadas e angulação para diferentes alturas

Essa configuração de posicionamento da placa, conforme Figura 5.15 foi determinada como padrão, tanto para o cadastro automático com busca por Template Matching, quanto para o alinhamento do robô colaborativo.

O robô consegue se guiar pelo ponto central da placa, sem variar o $Z$, já que não temos esse valor. E a orientação do $Rz$, ângulo do terceiro pulso do robô, onde está acoplada a câmera e a garra, foi obtido por meio deste processamento inicial e a adição da extração de informações do retângulo formado no contorno da placa. O ângulo que indica alinhamento com garra do robô é o de 90 graus, conforme Figura 5.3.

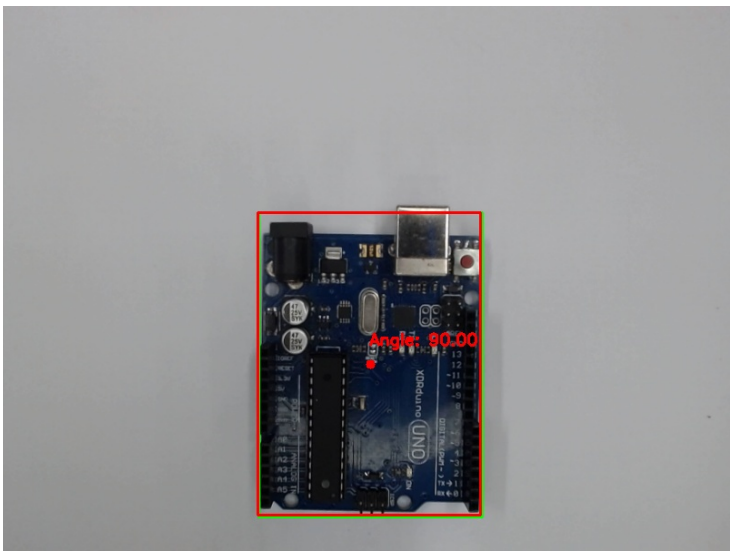



Figura 5.3: Identificação do ângulo (Rz)

## 5.3 Cadastro Automático da Placa

O algoritmo de cadastro automático da placa é composto pelo processamento de image utilizado nas etapas anteriores, aplicação do modelo pré-treinado FastSAM para segmentar os componentes da placa e utilização do Template Matching para analisar correlação da nova placa que está sendo inspecionada com a que foi cadastrada como padrão.

A etapa de cadastro é apenas uma foto da placa que vai servir como base para as inspeções automáticas seguintes. Todas essas etapas o robô colaborativo consegue executar com o algoritmo desenvolvido.

1. Robô vai até o ponto de cadastro;
2. Câmera captura a imagem de cadastro;
3. Inicia-se o novo fluxo de inspeção;
4. Robô se alinha na configuração que a nova placa chega no campo de visão;
5. A nova placa é inspecionada com o algoritmo de correlação com base no Template Matching;
6. Robô faz a separação da placa.

Nesse fluxo há duas etapas, na primeira, o algoritmo usa o FastSAM para segmentar a placa padrão e guardar os resultados referentes aos cortes dos componentes da placa, conforme Figura 5.4 e 5.5.

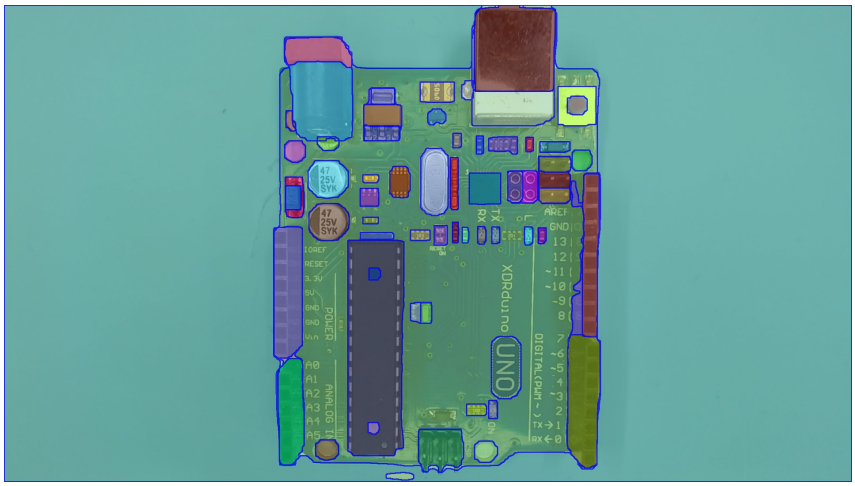



Figura 5.4: Componentes eletrônicos segmentados à uma altura de 11cm


Figura 5.5: Templates dos componentes gerados a partir da segmentação da PCB

Conforme o teste realizado para a segmentação para diferentes alturas, a quantidade de componentes segmentados, referentes às alturas 11cm, 15cm, 17.5cm e 20cm, pode ser vista na Tabela 5.1.

Tabela 5.1: Resultados de segmentação para diferentes alturas

| **Altura (cm)** | **Identificação** | **Número de Componentes** |
|---|---|---|
| 11 | Placa | 66 |
| 15 | Placa | 40 |
| 17.5 | Placa | 28 |
| 20 | Placa | 27 |

De acordo com a Tabela 5.1, para as distâncias consideradas, o algoritmo manteve a capacidade de segmentar os componentes principais da placa, assim como para todas essas alturas, ele conseguiu identificar a placa total, apresentando um desempenho consistente na identificação de regiões maiores e de maior contraste com o fundo. No entanto, foi possível perceber que componentes menores ou com cores similares à base da placa apresentaram dificuldades na segmentação, o que pode estar relacionado à percepção de detalhes pela câmera para distâncias além de $15cm$. Esses aspectos são importantes para compreender as limitações e os comportamentos do algoritmo sob diferentes condições de altura.

### 5.3.1 Identificação dos Componentes da PCB

O algoritmo de identificação dos componentes busca o padrão cadastrado nas novas placas, ao encontrar, gera um índice de correlação que neste caso é em média 100%, devido o alinhamento refinado do robô com a placa. Por este motivo, a metodologia propõe que o robô seja guiado pela visão para se alinhar à placa para que depois seja aplicado o algoritmo de identificação.

O resultado da correlação entre esses componentes aplicados nas novas placas, quando o robô está devidamente alinhado à elas pode ser visto nas Figuras 5.6, 5.7, 5.8 e 5.9. Vale ressaltar que os testes em

diferentes alturas é para analisar o algoritmo de identificação que é único para todas essas imagens, mas para cada altura, necessita-se que seja realizado o cadastro inicial, tornando-a padrão para aquela altura de inspeção.

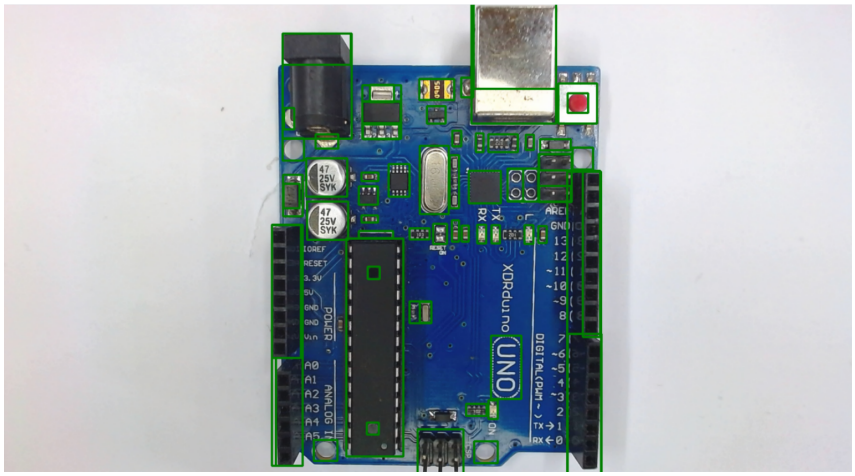


Figura 5.6: Aplicação dos templates dos componentes segmentados na PCB a 11cm de distância

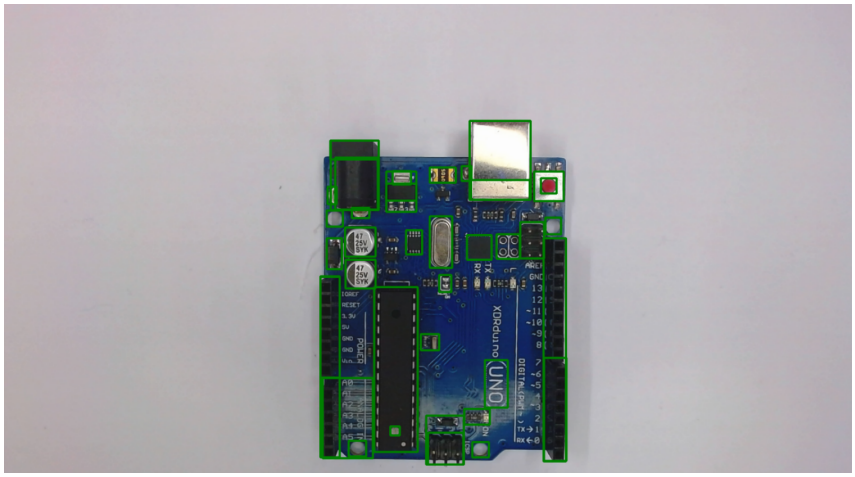


Figura 5.7: Aplicação dos templates dos componentes segmentados na PCB a 15cm de distância

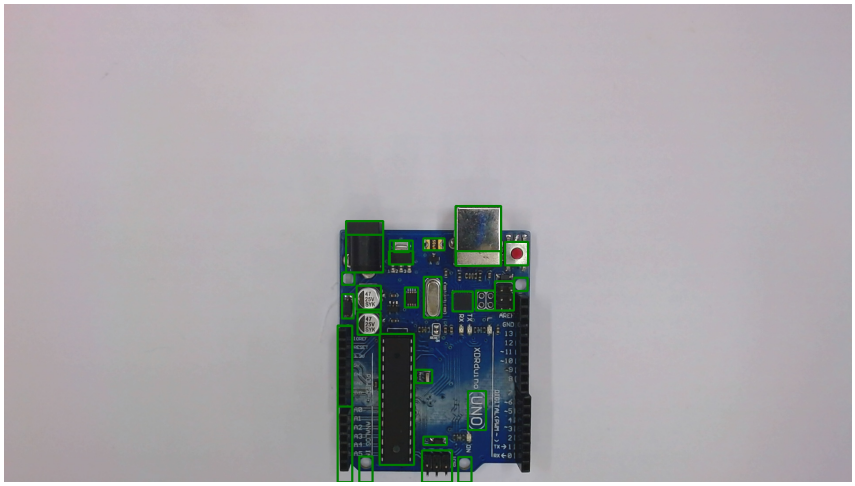


Figura 5.8: Aplicação dos templates dos componentes segmentados na PCB a 17.5cm de distância

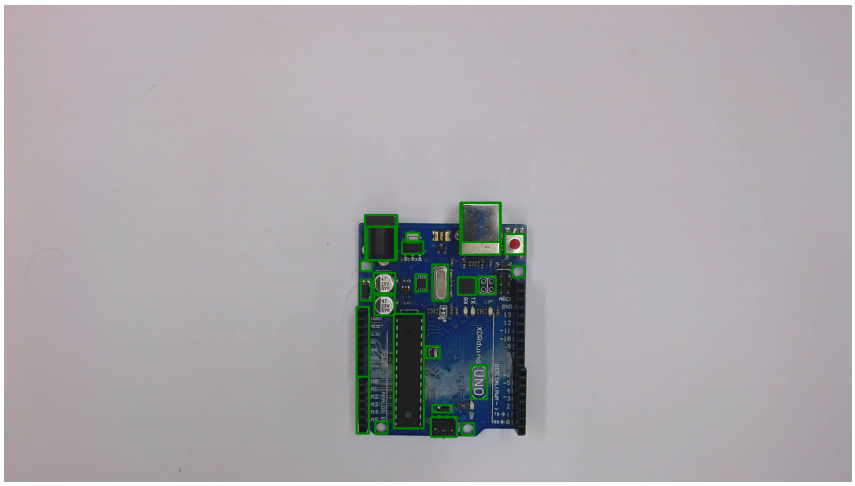


Figura 5.9: Aplicação dos templates dos componentes segmentados na PCB a 20cm de distância

### 5.3.2 Métricas de Avaliação da Segmentação

As métricas de avaliação fornecem uma análise detalhada sobre a qualidade da segmentação automática, Figura (a) de 5.11, em relação a *Ground Truth* (anotação de referência), Figura (b) de 5.11. Vale ressaltar que para o cálculo, a visualização e análise das métrica de segmentação, utilizou-se como base imagens tiradas à uma altura de 17.5cm, sendo a segmentação manual representada pela Figura 5.10, que simula o processo de rotulagem manual dos componentes, onde o operador precisa clicar em cada ponto ao redor do componente a ser segmentado.

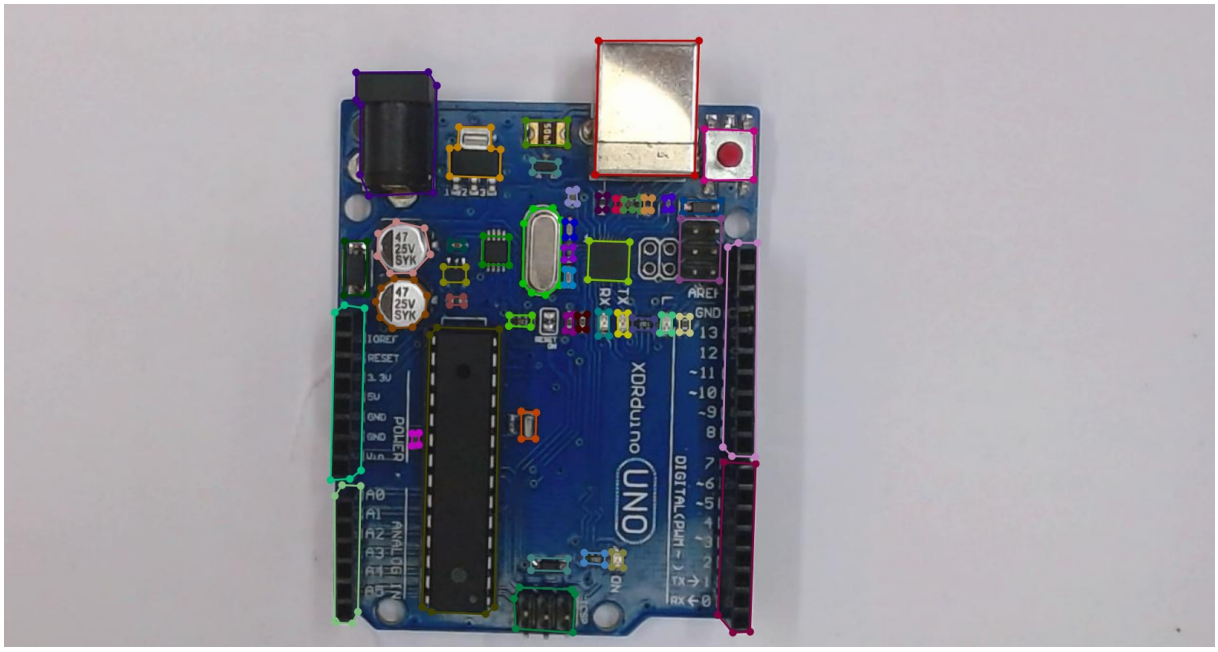


Figura 5.10: Processo manual de anotação de componentes de uma PCB

(a) Segmentação Manual de referência (*Ground Truth*) (b) Segmentação Automática

Figura 5.11: Visualização das máscaras da segmentação manual e da automática à distância de 17.5cm

A métricas aplicadas, para a avaliação da qualidade da segmentação automática com base na segmentação manual, são: Precisão, Revocação, Interseção sobre União (IoU), F1-Score, Acurácia, Verdadeiros Positivos (VP), Falsos Positivos (FP), Falsos Negativos (FN) e Verdadeiros Negativos (TN). Essas métricas são calculadas da seguinte forma:

- **Precision (Precisão)**: Mede a proporção de pixels preditos como segmento que são corretos. Valores mais altos indicam menos falsos positivos.

$$\text{Precision} = \frac{VP}{VP + FP} \tag{5.1}$$

- **Recall (Revocação)**: Mede a proporção de pixels pertencentes ao segmento que foram corretamente identificados. Valores altos indicam menos falsos negativos.

$$\text{Recall} = \frac{VP}{VP + FN} \tag{5.2}$$

- **Intersection over Union (IoU)**: Representa a interseção entre a predição e a Ground Truth, normalizada pela união de ambas. Indica o nível de sobreposição entre as duas máscaras.

$$\text{IoU} = \frac{VP}{VP + FP + FN} \tag{5.3}$$

- **F1-Score**: Combina precisão e recall em uma única métrica harmônica, indicando o equilíbrio entre ambos.

$$F1\text{-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.4}$$

- **Accuracy (Acurácia)**: Mede a proporção total de pixels corretamente classificados, considerando tanto o segmento quanto o fundo.

$$\text{Accuracy} = \frac{VP + TN}{VP + FP + FN + TN} \tag{5.5}$$

Essas fórmulas permitem avaliar diferentes aspectos do desempenho da segmentação. Por exemplo, altos valores de revocação indicam que o modelo está identificando a maioria dos segmentos presentes na *Ground Truth*, enquanto a precisão reflete a capacidade de evitar erros ao predizer segmentos irreais. Aplicando essas fórmulas na avaliação da segmentação automática em comparação à placa base (*Ground Truth*, obteve-se os resultados da Tabela 5.2. O processo de comparação necessitou da binarização das duas segmentações para que fosse possível separar os pixels do fundo da imagem com os segmentados, onde há componentes, para assim, obter o resultado, isto pode ser visualizado na Figura 5.12.

Tabela 5.2: Métricas de avaliação da segmentação automática

| **Métrica** | **Valor** |
|---|---|
| Precisão (%) | 83.77 |
| Revocação (%) | 90.43 |
| Interseção sobre União (IoU) (%) | 76.95 |
| F1-Score (%) | 86.97 |
| Acurácia (%) | 98.69 |
| Verdadeiros Positivos (VP) | 90417 |
| Falsos Positivos (FP) | 17516 |
| Falsos Negativos (FN) | 9571 |
| Verdadeiros Negativos (TN) | 1956096 |

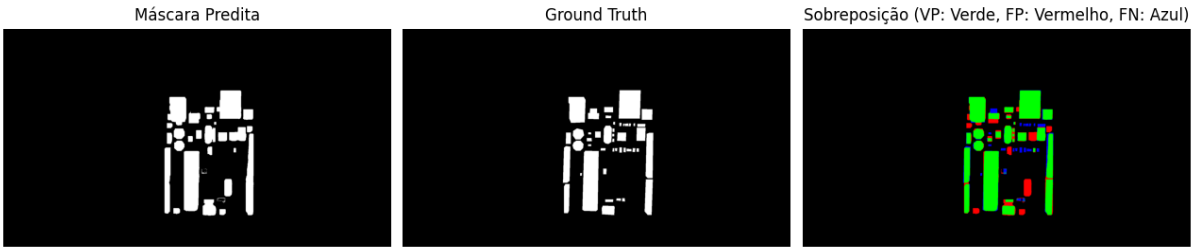



Figura 5.12: Comparação da placa segmentada automaticamente com a manual

A seguir estão as interpretações de cada métrica e sua relação com a imagem visualizada:

- **Precisão (83.77%)**: Representa a proporção de pixels corretamente identificados como segmento em relação ao total de pixels preditos. Um valor de 83.77% indica que o modelo possui uma boa capacidade de evitar falsos positivos (em vermelho na imagem), embora ainda existam alguns casos onde áreas não pertencentes ao segmento foram classificadas incorretamente.

- **Revocação (90.43%)**: Mede a proporção de pixels verdadeiramente segmentados que o modelo conseguiu identificar corretamente. Um *recall* de 90.43% demonstra que o modelo conseguiu capturar a maior parte do segmento verdadeiro, o que é evidenciado pela predominância de áreas verdes na Figura 5.12 (verdadeiros positivos).

- **IoU (76.95%)**: O *Intersection over Union* é a métrica que avalia a interseção entre a predição e a *Ground Truth* em relação à união de ambos. Com um IoU de 76.95%, pode-se inferir que há uma boa sobreposição entre as áreas segmentadas corretamente (verde), mas ainda há espaço para melhorias nas bordas ou áreas mais difíceis.

- **F1-Score (86.97%)**: Combina a precisão e a revocação para dar uma visão equilibrada do desempenho do modelo de segmentação. Com um F1-Score de 86.97%, o modelo demonstra um bom equilíbrio entre evitar falsos positivos e capturar os verdadeiros positivos.

- **Accuracy (98.69%)**: Reflete a proporção de pixels corretamente classificados (tanto segmento quanto fundo) em relação ao total de pixels. A alta acurácia é suportada pelo grande número de verdadeiros negativos (fundo correto) na imagem, mostrado pela predominância de preto na Figura 5.12.

Analisando novamente a forma e cores da sobreposição da placa segmentada com a manual, 5.14, tem-se:

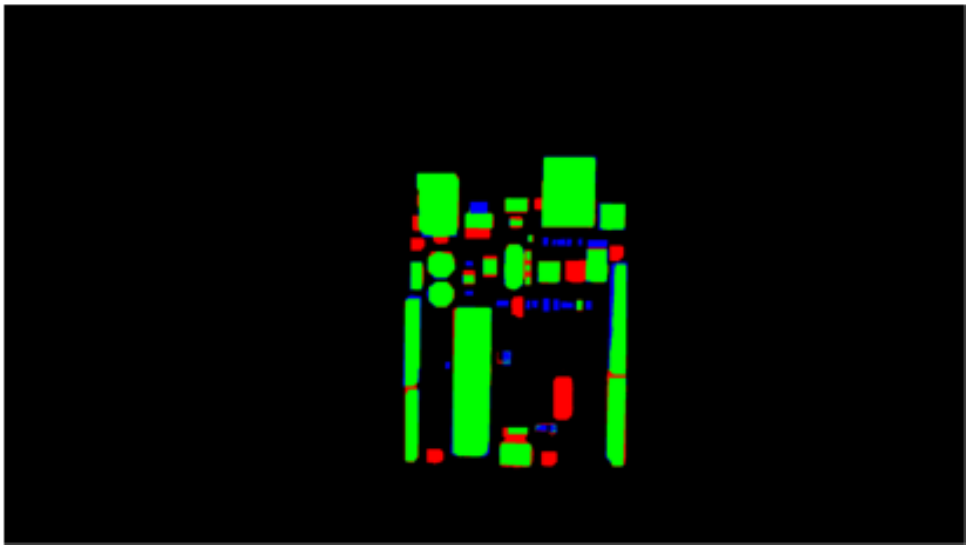


Figura 5.13: Resultado da sopreposição

- **Verdadeiros Positivos (VP):** 90417 – são os pixels corretamente identificados como segmento, destacados em **verde** na imagem;
- **Falsos Positivos (FP):** 17516 – representam os pixels incorretamente identificados como segmento, destacados em **vermelho**;
- **Falsos Negativos (FN):** 9571 – são os pixels que deveriam ser identificados como segmento, mas foram classificados como fundo, mostrados em **azul**;
- **Verdadeiros Negativos (TN):** 1956096 – representam os pixels corretamente classificados como fundo, que são a maioria, explicando a alta acurácia.

A combinação dessas métricas e a análise visual da sobreposição demonstram que o modelo tem um bom desempenho geral, mas pode melhorar nas bordas, onde aparecem os falsos positivos (FP) e falsos negativos (FN), otimizando ainda mais a correspondência com a imagem de base padrão *Ground Truth*.

### 5.3.3 Movimentação do Robô

O processo da movimentação do robô em relação às coordenadas $(x, y)$ e ao ângulo $Rz$ tem como objetivo avaliar o comportamento do sistema durante o alinhamento de posição e angular guiado pela visão computacional, além de observar as variações nos erros ao longo das iterações.

A Tabela 5.3 apresenta os registros das coordenadas detectadas pelo sistema e os erros correspondentes ao longo das iterações para o alinhamento nas coordenadas $(x, y)$. Os erros diminuem gradualmente, indicando o refinamento do algoritmo na correção da posição do robô até atingir os critérios de tolerância pré-definidos.

Tabela 5.3: Leitura dos resultados do alinhamento nas coordenadas $(x, y)$

| **Iteração** | **Coordenadas** $(x, y)$ | **Erro** $(x, y)$ |
|---|---|---|
| 1 | (955, 648) | (-0.0010, 0.1806) |
| 2 | (922, 678) | (0.0161, 0.1528) |
| 3 | (900, 708) | (0.0276, 0.1250) |
| 4 | (877, 732) | (0.0396, 0.1028) |

Os resultados indicam que, após cinco iterações, o alinhamento de posição foi concluído com sucesso, ou seja, atingiu o centro da placa alinhada ao centro da garra para uma melhor "pega"do robô, atingindo o critério de tolerância definido.

O processo de movimentação para o alinhamento angular requer que o sistema inicie alinhando primeiramente os eixos $(x, y)$ e posteriormente é realizado o ajuste do ângulo $Rz$, para evitar possíveis distorções de ângulo e para simplificar o processo fazendo o cálculo do ângulo apenas uma vez. A Tabela 5.4 resume as informações detectadas pela câmera, incluindo o ângulo inicial, o erro angular e o ângulo desejado para o robô.

Tabela 5.4: Leituras dos resultados do alinhamento angular $R_z$

| **Iteração** | **Ângulo Inicial (°)** | **Erro Angular (°)** | **Ângulo Atual (°)** | **Ângulo Desejado (°)** | **Direção** |
|---|---|---|---|---|---|
| 1 | 60.87 | 29.13 | -116.10 | -142.83 | Horário |
| 2 | 60.95 | 29.05 | -142.84 | -142.75 | Anti-horário |
| 3 | 43.06 | 46.94 | -115.87 | -160.64 | Horário |
| 4 | 49.40 | 40.60 | -113.03 | -154.30 | Horário |

Os registros mostram que o sistema ajustou o ângulo com sucesso em todas as iterações, utilizando a direção correta de rotação e alcançando os valores desejados dentro da tolerância estabelecida.

Após o alinhamento das coordenadas e ângulo, foi realizada uma movimentação no eixo $Z$ para que o *cobot* pudesse realizar o *Pick-and-Place* da placa para o local de destino.

Os registros do posicionamento guiado do robô pela câmera indicam que o sistema de controle do robô é eficaz no alinhamento posicional e angular. O alinhamento em $(x, y)$ apresentou convergência rápida, enquanto o ajuste do ângulo Rz demonstrou precisão ao alcançar os valores desejados. A movimentação no eixo $Z$ foi consistente, reforçando a confiabilidade do sistema.

Os dados sugerem que o sistema pode ser aprimorado para lidar com erros iniciais maiores, e demonstra robustez em ambientes controlados.

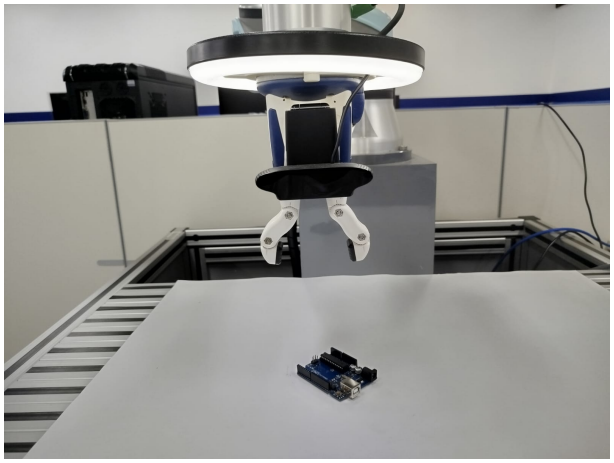
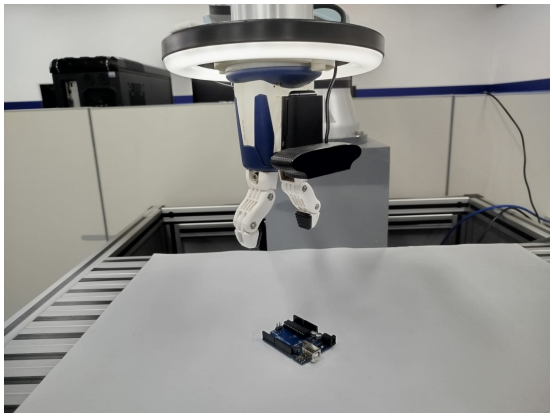
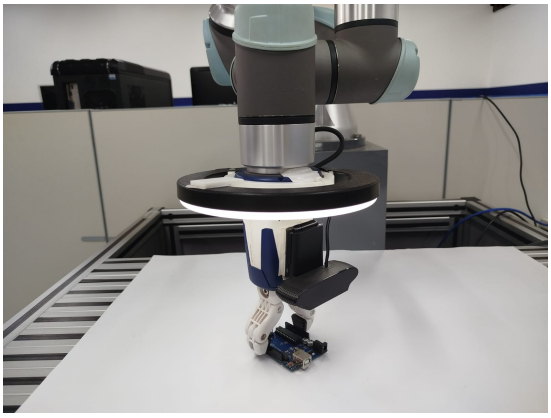


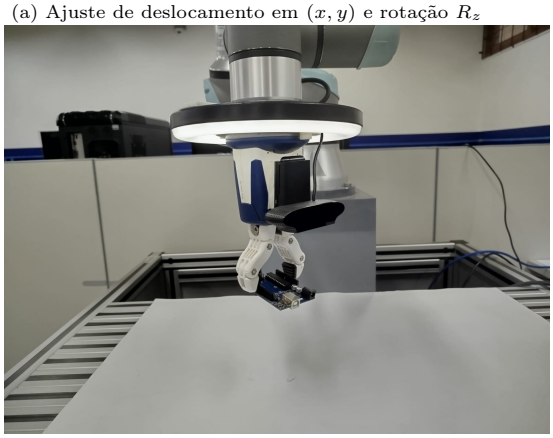
Figura 5.14: Posição inicial do *cobot* e placa deslocada e rotacionada



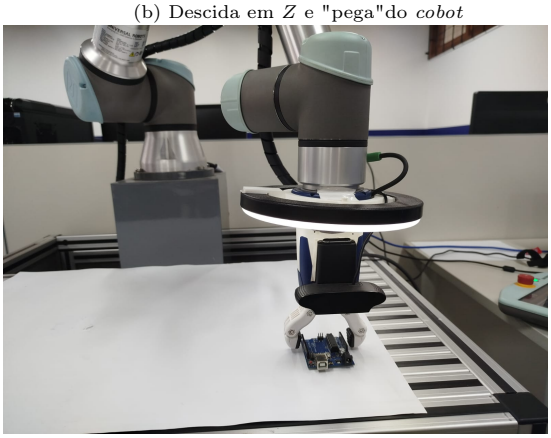
(a) Ajuste de deslocamento em $(x, y)$ e rotação $R_z$



(b) Descida em $Z$ e "pega"do *cobot*



(c) Subida em $Z$ com "pega"consistente do *cobot*



(d) Movimentação para o destino final da placa

Figura 5.15: Movimentação do *cobot* guiado pela visão computacional

Link do vídeo da movimentação do *cobot* durante o processo: https://drive.google.com/drive/folders/19aUMZK2_ECUb2zKG5c2d_Bk_bFHi1sbm?usp=drive_link.

## 5.4 Conclusão

Este capítulo apresentou os resultados dos testes realizados a partir do algoritmo de visão computacional para detecção da PCB em diferentes alturas, determinação do alinhamento do robô de acordo com o posicionamento e a rotação da placa dentro do campo de visão, segmentação da placa e identificação de componentes em alturas variadas, assim como a apresentação das métricas para análise da segmentação, além da movimentação do robô apresentando as leituras do algoritmo de posicionamento e ângulo. Na próxima seção, serão comentadas as conclusões a partir desses testes e resultados.

# Capítulo 6

# Conclusão

Neste trabalho, foi proposta a utilização de visão computacional e robótica colaborativa para automatizar um processo industrial de inspeção de placas de circuitos impressos (PCIs). Para isso, foram desenvolvidos diversos módulos, abrangendo desde a programação do robô colaborativo, a escolha dos equipamentos (câmera e iluminação), fabricação de um suporte para acoplar a iluminação no braço robótico, até a aplicação de algoritmos de visão computacional capazes de orientar o robô e realizar a inspeção de placas em diferentes posições, orientações e alturas, dentro do campo de visão da câmera acoplada no robô.

Os resultados apresentados evidenciam os benefícios do sistema em realizar a segmentação automática de componentes das PCIs diante do cenário sem dados anotados. Por meio das métricas de avaliação, como precisão (83,77%), revocação (90,43%), IoU (76,95%), F1-Score (86,97%) e acurácia de pixel (98,69%), foi possível validar a eficiência do modelo em identificar os componentes e diferenciá-los do fundo. Contudo, ainda se observa a necessidade de melhorias no algoritmo de segmentação para lidar com componentes muito pequenos ou de cores similares à base da placa, que apresentaram taxas de falsos negativos mais altas.

A análise dos registros de posicionamento do *cobot* demonstrou a eficácia do sistema em alinhar a ferramenta robótica em coordenadas $(x, y)$ e no ângulo $R_z$, garantindo precisão nos movimentos para a realização das tarefas de inspeção. Durante o alinhamento de posição, os erros de deslocamento em $(x, y)$ diminuíram progressivamente ao longo das iterações, convergindo para valores dentro da tolerância estabelecida de alinhamento ao centro da placa. O alinhamento angular também apresentou exatidão, com ajustes precisos no ângulo $R_z$ detectado pela câmera, corrigindo desvios e garantindo que o *cobot* estivesse com sua orientação correta para a execução da "pega"robótica.

Os resultados destacam a robustez do sistema no controle de posição e na orientação angular, mesmo em cenários onde o erro inicial é significativo (quando a placa está distante da garra), visto que como o robô é guiado pela visão e esta toma como base os pixels da imagem, quanto maior a distância, maior o erro estimado, por isso o *cobot* necessita dar mais passos para refinar esse alinhamento. Além disso, a movimentação no eixo $Z$ mostrou consistência, com ajustes precisos para posicionar a ferramenta em alturas adequadas para a inspeção e manipulação dos componentes, tomando como base, a altura da base

para a ponta da garra e a altura da placa.

Combinando os resultados de posicionamento com os de segmentação, conclui-se que o sistema proposto é capaz de integrar visão computacional e controle robótico de maneira eficiente e precisa. A segmentação automática dos componentes das placas, mesmo em condições desafiadoras de posicionamento, altura e iluminação, aliada à capacidade do *cobot* de corrigir erros de posição e angular guiado pela visão, evidencia o potencial do sistema para aplicações industriais.

Uma das principais contribuições deste trabalho é a eliminação da necessidade de *datasets* rotulados para treinamento do modelo de segmentação, além de automatizar o cadastro de componentes de forma robusta para diferentes alturas. Este cadastro inteligente foi implementado com base no modelo FastSAM, combinado com a técnica *Template Matching*, permitindo uma análise totalmente automatizada e flexível. Em adição a isto, o aumento da altura da câmera para ampliar o campo de visão demonstrou impacto na segmentação de microcomponentes, mas o sistema continuou apresentando resultados satisfatórios para a maioria dos itens.

Este trabalho também demonstrou a viabilidade de integrar sistemas de visão computacional com robótica colaborativa em ambientes industriais. A abordagem proposta proporciona uma solução escalável e adaptável, alinhada aos princípios da Indústria 4.0. Além disso, promove um ambiente colaborativo seguro, onde robôs podem trabalhar lado a lado com operadores humanos, aumentando a eficiência, qualidade e flexibilidade do processo produtivo.

Como trabalho futuro, sugere-se explorar técnicas de aprendizado mais avançadas, como métodos de atenção contextual e *transformers*, para melhorar a segmentação de componentes complexos ou pouco contrastantes. Também seria interessante avaliar o impacto da iluminação dinâmica e de diferentes materiais no desempenho do sistema. E o aprimoramento dos algoritmos de controle para reduzir ainda mais o tempo de convergência durante o alinhamento de posição e angulação.

# Referências Bibliográficas


B. Jähne, *Digital Image Processing, 5th revised and extended edition*, 5th ed. Springer, 2002.

A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," 2023.

Universal Robots, "Produtos," 2024, acesso em: 29 nov. 2024. [Online]. Available: https://www.universal-robots.com/pt/produtos/

——, "Ur3/cb3: Manual do usuário," https://www.universal-robots.com/download/manuals-cb-series/user/ur3/315/user-manual-ur3-cb-series-sw315-portuguese-pt/, 2021, acessado em 20/07/2023.

G. Boschetti and T. Sinico, "Performance comparison of two architectures of 6r articulated robots," *Machines*, vol. 11, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2075-1702/11/2/306

A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," 2019.

R. Gonzalez and R. Woods, *Digital Image Processing*. Pearson Education, 2008.

SDU Robotics, "Ur rtde python library - examples," 2024, acesso em: 29 nov. 2024. [Online]. Available: https://sdurobotics.gitlab.io/ur_rtde/examples/examples.html

Logitech, "Logitech c920e business webcam," 2024, acesso em: 29 nov. 2024. [Online]. Available: https://www.logitech.com/pt-br/products/webcams/c920e-business-webcam.960-001401.html

S. D. Han, S. W. Feng, and J. Yu, "Toward fast and optimal robotic pick-and-place on a moving conveyor," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 446–453, 2020.

P. Francesco and G. G. Paolo, "Aura: An example of collaborative robot for automotive and general industry applications," *Procedia Manufacturing*, vol. 11, pp. 338–345, 2017, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

Z. Pan, Z. Jia, K. Jing, Y. Ding, and Q. Liang, "Manipulator package sorting and placing system based on computer vision," in *2020 Chinese Control And Decision Conference (CCDC)*, 2020, pp. 409–414.

M. Gualtieri and R. Platt, "Robotic pick-and-place with uncertain object instance segmentation and shape completion," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1753–1760, 2021.

H. L. Vieira, E. Wajnberg, A. T. Beck, and M. Martins da Silva, "Reliable motion planning for parallel manipulators," *Mechanism and Machine Theory*, vol. 140, pp. 553–566, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X19308894

H. L. Vieira, J. V. C. Fontes, A. T. Beck, and M. M. da Silva, "Reliable and Failure-Free Workspaces for Motion Planning Algorithms for Parallel Manipulators Under Geometrical Uncertainties," *Journal of Computational and Nonlinear Dynamics*, vol. 14, no. 2, 01 2019, 021005. [Online]. Available: https://doi.org/10.1115/1.4042015

A. Nourmohammadi, M. Fathi, and A. H. Ng, "Balancing and scheduling human-robot collaborated assembly lines with layout and objective consideration," *Computers  Industrial Engineering*, vol. 187, p. 109775, 2024.

M. Doyle-Kent and P. Kopacek, "Adoption of collaborative robotics in industry 5.0. an irish industry case study," *IFAC-PapersOnLine*, vol. 54, no. 13, pp. 413–418, 2021, 20th IFAC Conference on Technology, Culture, and International Stability TECIS 2021.

V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

J. Male and U. Martinez-Hernandez, "Deep learning based robot cognitive architecture for collaborative assembly tasks," *Robotics and Computer-Integrated Manufacturing*, vol. 83, p. 102572, 2023.

Y. Yang and H. Kang, "An enhanced detection method of pcb defect based on improved yolov7," *Electronics*, vol. 12, no. 9, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/9/2120

M. Lakhe and P. Shinghare, "Stitching micro images of pcb's using computer vision system," in *2022 International Conference on Signal and Information Processing (IConSIP)*, 2022, pp. 1–5.

R. Aarthi. and G. Rishma., "A vision based approach to localize waste objects and geometric features exaction for robotic manipulation," *Procedia Computer Science*, vol. 218, pp. 1342–1352, 2023, international Conference on Machine Learning and Data Engineering. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050923001138

B. Ioller, "Bottle sorting by cnn and physical features extraction via robotic arm," Nov. 2019, working paper or preprint. [Online]. Available: https://hal.science/hal-02343201

J. Kim, O. Nocentini, M. Scafuro, R. Limosani, A. Manzi, P. Dario, and F. Cavallo, "An innovative automated robotic system based on deep learning approach for recycling objects," 07 2019.

H. Wang, J. Xie, X. Xu, and Z. Zheng, "Few-shot pcb surface defect detection based on feature enhancement and multi-scale fusion," *IEEE Access*, vol. 10, pp. 129 911–129 924, 2022.

J. Li, J. Gu, Z. Huang, and J. Wen, "Application research of improved yolo v3 algorithm in pcb electronic component detection," *Applied Sciences*, vol. 9, no. 18, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/18/3750

J. Wang and H. Li, "A comprehensive review of deep learning-based pcb defect detection," *IEEE Access*, vol. 11, pp. 12 345–12 359, 2023.

S. Khan and R. Ahmed, "Detection and classification of pcb defects using deep learning methods," *Springer Neural Computing*, vol. 15, pp. 23–34, 2023.

T. Smith and L. Zhang, "Flaw detection in pcb using deep learning and image processing," *IEEE Transactions on Electronics*, vol. 19, pp. 12–24, 2022.

F. Ahmed and P. Wang, "Comparative analysis of existing deep learning techniques for automatic pcb defect detection," *IEEE Access*, vol. 11, pp. 12 998–13 012, 2023.

X. Chen and J. Li, "Research on robotic arm based on yolo," *IEEE Transactions on Industrial Applications*, vol. 58, pp. 23–34, 2023.

W. Zhang and Y. Liu, "Sc-yolo: Provide application-level recognition and perception," *Springer Neural Computing*, vol. 35, pp. 1123–1135, 2023.

C. Wang and M. Zhao, "Yolo-cea: A real-time industrial defect detection method based on contextual enhancement and attention," *IEEE Access*, vol. 11, pp. 14 123–14 135, 2023.

J. Li and M. Chen, "Yolov6: A single-stage object detection framework dedicated to industrial applications," *IEEE Transactions on Automation Science and Engineering*, vol. 20, pp. 1024–1035, 2023.

R. Kumar and A. Singh, "Yolov8-tdd: An optimized yolov8 algorithm for targeted defect detection in pcbs," *Journal of Manufacturing Processes*, vol. 72, pp. 45–58, 2024.

P. Garcia and M. Lopez, "An fpga-based yolov5 accelerator for real-time industrial vision applications," *Micromachines*, vol. 14, p. 2345, 2023.

R. Patel and K. Sharma, "A survey on industrial defect detection using deep learning," *Journal of Intelligent Manufacturing*, vol. 34, pp. 122–140, 2023.

X. Wang and W. Yu, "Adaptive yolo for industrial surface defect detection," *IEEE Transactions on Automation Science and Engineering*, vol. 21, pp. 450–465, 2024.

L. Nguyen and T. Pham, "Pcb defect detection using transformer-based networks," *IEEE Access*, vol. 11, pp. 15 000–15 015, 2023.

R. Shanmugamani, *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras*. Packt Publishing Ltd, 2018.

L. G. Shapiro and G. C. Stockman, *Computer Vision*. Prentice-Hall, 2001.

N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

D. Guo, Y. Pei, K. Zheng, H. Yu, Y. Lu, and S. Wang, "Degraded image semantic segmentation with dense-gram networks," *IEEE Transactions on Image Processing*, vol. 29, pp. 782–795, 2020.

J. Yi, P. Wu, M. Jiang, Q. Huang, D. J. Hoeppner, and D. N. Metaxas, "Attentive neural cell instance segmentation," *Medical Image Analysis*, vol. 55, pp. 228–240, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841518308442

C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Advances in neural information processing systems*, vol. 27, 2014.

C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 951–958.

L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, no. 1. Lille, 2015, pp. 1–30.

R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial intelligence review*, vol. 18, pp. 77–95, 2002.

C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," *Advances in neural information processing systems*, vol. 30, 2017.

B. Hariharan and R. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3018–3027.

H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, "Dreaming to distill: Data-free knowledge transfer via deepinversion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8715–8724.

H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3588–3597.

Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning – a comprehensive evaluation of the good, the bad and the ugly," 2020.

R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley, 2009.

D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 200, no. 1140, pp. 269–294, 1978.

N. Lucci, A. Monguzzi, A. M. Zanchettin, and P. Rocco, "Workflow modelling for human–robot collaborative assembly operations," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102384, 2022.

A. Koszulinski, J. Sandoval, T. Vendeuvre, S. Zeghloul, and M. A. Laribi, "Comanipulation robotic platform for spine surgery with exteroceptive visual coupling: development and experimentation," *Journal of Medical Devices*, vol. 16, no. 4, p. 041002, 2022.

Universal Robots, "A história da robótica: Dos autômatos antigos aos cobots e outros robôs modernos," 2024, acessado em 18 de novembro de 2024. [Online]. Available: https://www.universal-robots.com/br/blog/a-hist%C3%B3ria-da-rob%C3%B3tica-dos-aut%C3%B4matos-antigos-aos-cobots-e-outros-rob%C3%B4s-modernos/

A. Baratta, A. Cimino, M. G. Gnoni, and F. Longo, "Human robot collaboration in industry 4.0: a literature review," *Procedia Computer Science*, vol. 217, pp. 1887–1895, 2023, 4th International Conference on Industry 4.0 and Smart Manufacturing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922024747

Fournier, D. Kilgus, A. Landry, B. Hmedan, D. Pellier, H. Fiorino, and C. Jeoffrion, "The impacts of human-cobot collaboration on perceived cognitive load and usability during an industrial task: An exploratory experiment," *arXiv preprint arXiv:2305.18913*, 2023. [Online]. Available: https://arxiv.org/abs/2305.18913

M. Neves, L. Duarte, and P. Neto, "The role of deep learning in collaborative robotics: Challenges and opportunities," *arXiv preprint arXiv:2403.05306*, 2024. [Online]. Available: https://arxiv.org/abs/2403.05306

J. Heredia, M. A. Cabrera, J. Tirado, V. Panov, and D. Tsetserukou, "Cobotgear: Interaction with collaborative robots using wearable optical motion capturing systems," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 1584–1589.

F. T. S. Pereira and E. O. Simonetto, "Robótica colaborativa na indústria 4.0, sua importância e desafios," *Interface Tecnológica*, vol. 18, no. 2, pp. 748–759, 2021. [Online]. Available: https://revista.fatectq.edu.br/interfacetecnologica/article/view/1298

A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, p. 957–975, 2018.

E. Matheson, R. Minto, E. G. G. Zampieri, M. Faccio, and G. Rosati, "Human–robot collaboration in manufacturing applications: A review," *Robotics*, vol. 8, no. 4, 2019.

M. Javaid, A. Haleem, R. P. Singh, S. Rab, and R. Suman, "Significant applications of cobots in the field of manufacturing," *Cognitive Robotics*, vol. 2, pp. 222–233, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667241322000209

J. E. Colgate, W. Wannasuphoprasit, and M. A. Peshkin, "Cobots: Robots for collaboration with human operators," vol. Dynamic Systems and Control, pp. 433–439, 11 1996. [Online]. Available: https://doi.org/10.1115/IMECE1996-0367

K. R. Guerin, C. Lea, C. Paxton, and G. D. Hager, "A framework for end-user instruction of a robot assistant for manufacturing," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6167–6174.

R. Müller, M. Vette, and A. Geenen, "Skill-based dynamic task allocation in human-robot-cooperation with the example of welding application," *Procedia Manufacturing*, vol. 11, pp. 13–21, 2017, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978917303177

L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda, "Cyber-physical systems in manufacturing," *CIRP Annals*, vol. 65, no. 2, pp. 621–641, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007850616301974

N. Nikolakis, V. Maratos, and S. Makris, "A cyber physical system (cps) approach for safe human-robot collaboration in a shared workplace," *Robotics and Computer-Integrated Manufacturing*, vol. 56, pp. 233–243, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584517302168

S. Haddadin, *Towards safe robots: approaching Asimov's 1st law*. Springer, 2013, vol. 90.

A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, and P. Fraisse, "Collaborative manufacturing with physical human–robot interaction," *Robotics and Computer-Integrated Manufacturing*, vol. 40, pp. 1–13, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584515301769

M. Pearce, B. Mutlu, J. Shah, and R. Radwin, "Optimizing makespan and ergonomics in integrating collaborative robots into manufacturing processes," *IEEE transactions on automation science and engineering*, vol. 15, no. 4, pp. 1772–1784, 2018.

R. Meneguelli, "Utilização de visão computacional e robótica colaborativa para a coleta seletiva de lixo," 2022.

S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: https://books.google.com.br/books?id=-PwLBAAAQBAJ

Universal Robots, "Programming collaborative robots: Tools and applications," 2023, acesso em: 29 nov. 2024. [Online]. Available: https://www.universal-robots.com/

——, *URScript Programming Guide*, 2023, acesso em: 29 nov. 2024. [Online]. Available: https://www.universal-robots.com/articles/ur/urscript/

——, "Simplificando a programação robótica com código g," 2020, acesso em: 29 nov. 2024. [Online]. Available: https://www.universal-robots.com/br/blog/simplificando-a-programac%C3%A3o-rob%C3%B3tica-com-c%C3%B3digo-g/

——, *URCaps SDK*, 2023, acesso em: 29 nov. 2024. [Online]. Available: https://www.universal-robots.com/download/software-e-series/support/urcaps-sdk/

——, *URCap SDK*, 2023, acesso em: 29 nov. 2024. [Online]. Available: https://www.universal-robots.com/plus/urcap-sdk/

——, "Benefícios dos robôs colaborativos," 2024, acesso em: 29 nov. 2024. [Online]. Available: https://www.universal-robots.com/br/sobre-a-universal-robots/rob%C3%B4s-colaborativos-benef%C3%ADcios-dos-cobots/