UNIVERSIDADE
BEIRA INTERIOR

Visão computacional como ferramenta de apoio ao controlo de qualidade e manipulação robotizada de frutas

Versão final após discussão

Estêvão Farias Vale Filho

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletromecânica
(2º ciclo de estudos)

Orientador: Prof. Doutor Pedro Miguel de Figueiredo Dinis Oliveira Gaspar
Co-orientador: Mestre Martim Lima de Aguiar

julho de 2024

Declaração de Integridade

Eu, Estêvão Farias Vale Filho, que abaixo assino, estudante com o número de inscrição M12295 de/o Engenharia Eletromecânica da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã _24_ /_07_ /_2024_

(assinatura conforme Cartão de Cidadão ou preferencialmente
assinatura digital no documento original se naquele mesmo formato)

Agradecimentos

Desejo expressar o meu mais profundo agradecimento ao meu orientador, Professor Doutor Pedro Miguel de Figueiredo Dinis Oliveira Gaspar, pela confiança e sabedoria que sempre transmitiu ao longo do trabalho desenvolvido nesta dissertação e por todas as oportunidades que me permitiram crescer enquanto pessoa e profissional.

Um agradecimento especial ao meu coorientador, Professor Martim Lima de Aguiar, ao Mestre Engenheiro Rodrigo Mendes Antunes e ao Doutor Nuno José Matos Pereira por toda a ajuda, disponibilidade e discussões construtivas desde o primeiro dia, as quais foram fundamentais para esta dissertação.

Agradeço aos meus camaradas de laboratório, Luan Matheus Capeletti Lang e Juan Angel Barreno Barreno, pelo companheirismo diário ao longo de todo o percurso do mestrado.

Agradeço ao meu pai, Estêvão Farias Vale, à minha mãe, Sarah Maria Monte das Oliveiras, e à minha irmã, Lara Maria Monte das Oliveiras Vale, por sempre me incentivarem e acreditarem no meu sucesso.

Aos meus avós, tios e madrinha, pelo apoio incondicional, independentemente do momento e da dor da saudade.

A todos os amigos que fiz nesta jornada, o meu mais profundo obrigado. Sem vocês, nada disso seria possível. O nome de todos estará para sempre marcado nas minhas memórias.

E, por fim, à minha companheira, Sarah de Castro, a pessoa que me apoiou incondicionalmente do início ao fim, que esteve ao meu lado nos momentos mais difíceis, os meus mais sinceros agradecimentos. Sem ti nada disso seria possível!

Resumo

O paradigma da utilização de novas tecnologias na agricultura e na indústria alimentar conduz a um aumento da produção e a uma potencial diminuição das perdas ao longo da cadeia de abastecimento. Porém, novos desafios surgem na modelação desta indústria, de modo a criar ambientes fiáveis com estas novas ferramentas.

Esta dissertação tem como objetivo estudar e aplicar duas dessas novas tecnologias, a visão computacional em conjunto com a robótica colaborativa, de modo a criar um sistema não destrutivo, ou seja, sem danificar o fruto, de fácil implementação e economicamente favorável para atuação em linhas de processamento de frutos e vegetais, auxiliando de maneira direta na deteção de falhas e controlo de qualidade.

Foi desenvolvido um sistema utilizando um Raspberry Pi 5 para captura de imagens dos frutos por meio de uma PiCamera módulo 3. As imagens foram enviadas a um braço robótico UR3e da Universal Robots via cabo Ethernet utilizando um código Python que integra funções desenvolvidas pela própria empresa e funções próprias desenvolvidas especificamente para esta dissertação.

Foram desenvolvidos quatro modelos de deteção de objetos utilizando o TensorFlow Object Detection API, convertendo-os posteriormente em TensorFlow Lite para detetar dois tipos de fruta (laranja e tomate) fazendo uso de técnicas de aprendizagem profunda. Cada fruto teve dois tipos de modelo, com variação de classes no modelo de tomate, porém com a mesma base de dados e com duas bases de imagens e número de classes diferentes para o modelo de laranja, o de laranja com 2 classes obteve 69,85% mAP, laranja 4 classes 46,37% mAP, tomate quatro classe 78,69% mAP e tomate duas classes 81,43% mAP para uma IoU de 0,5.

Por fim, criou-se uma área de trabalho retangular fiável para atuação do braço robótico em conjunto com a visão computacional. Após a realização de 640 testes de manipulação, obteve-se uma área fiável de 262 x 250 mm.

Palavras-chave

Robotica;Visão Computacional;Controlo de Qualidade; RaspBerry Pi; UR3e

Abstract

The paradigm of using new technologies in agriculture and the food industry is leading to an increase in production and a potential reduction in losses along the supply chain. However, new challenges arise in modelling this industry in order to create reliable environments with these new tools.

This dissertation aims to study and apply two of these new technologies, computer vision in conjunction with collaborative robotics, in order to create a non-destructive system, i.e. without damaging the fruit, that is easy to implement and economically favourable for use in fruit and vegetable processing lines, directly helping with fault detection and quality control.

A system was developed using a Raspberry Pi 5 to capture images of the fruit using a PiCamera module 3. The images were sent to a Universal Robots UR3e robotic arm via Ethernet cable using Python code that integrates functions developed by the company itself and its own functions developed specifically for this dissertation.

Four object detection models were developed using the TensorFlow Object Detection API and then converted into TensorFlow Lite to detect two types of fruit (oranges and tomatoes) using deep learning techniques. Each fruit had two types of model, with a variation of classes in the tomato model, but with the same database and with two different image bases and number of classes for the orange model, the orange with 2 classes obtained 69.85% mAP, orange 4 classes 46.37% mAP, tomato four classes 78.69% mAP and tomato two classes 81.43% mAP for an IoU of 0.5.

Finally, a reliable rectangular work area was created for the robotic arm to operate in conjunction with computer vision. After carrying out 640 manipulation tests, a reliable area of 262 x 250 mm was obtained.

Keywords

Robotics;Computer Vision;Quality Control; RaspBerry Pi; UR3e

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AP	Precisão Média
API	*Application Programming Interface*
AR	Recuperação Média
BGR	*Blue Green Red*
CAD	*Computer-aided design*
CNNs	Redes Neuronais Convolucionais
CPU	*Central Processing Unit*
CSI	*Camera Serial Input*
CUDA	*Compute Unified Device Architecture*
CV	*Computer Vision*
DBNs	*Deep Belief Network*
DCF	*Discriminative Correlation Filter*
DL	*Deep Learning*
DOF	*Degrees of freedom*
DUCNN	*Deep Convolutional Neural Network*
FAO	*The Food and Agriculture Organization*
FPS	*Frame per Second*
FRE	*Fin Ray Effect*
GANs	*Generative Adversarial Network*
GHz	*Gigahertz*
GPU	*Graphics Processing Unit*
GTX	*Giga Texel Shader Extreme*
HCI	*Human and Computer Interaction*
IA	Inteligência Artificial
IP	*Internet Protocol*
IoT	*Internet of Things*
IoU	Intersecção sobre União
ISP	*Internet Service Provider*
LSTMs	*Long Short-Term Memory*
ML	*Machine Learning*
MLPs	*Multilayer Perceptron*
mAP	*Mean Average Precision*
MS	Multiespectral
MS COCO	*Microsoft Common Objects in Context*
MSI	*Imagem Multiespectral*
PLC	*Programmable Logic Controllers*
RBFNs	*Radial Basis Function Network*
RBMs	*Restricted Boltzmann Machine*
RGB	*Red Green Blue*
RNNs	*Recurrent Neural Networks*
SGD	*Stochastic gradient descent*
SOMs	*Self-Organizing Map*
TCP	*Transmission Control Protocol*
TF	*TensorFlow*
TPU	*Thermoplastic Polyurethane Sheet*

Capítulo 1
Introdução

Neste capítulo apresenta-se a contextualização do trabalho desenvolvido, as motivações que conduziram a este estudo, os objetivos gerais e específicos que se esperam atingir com o presente trabalho, a metodologia adotada e a estrutura global desta dissertação.

1.1 Contextualização do trabalho desenvolvido

A busca pelo aumento da produtividade industrial tem-se intensificado, ano após ano. Esta procura por novas tecnologias a implementar em diversos setores é uma tendência que se manifesta desde os últimos anos da segunda revolução industrial. O primeiro setor a registar avanços tecnológicos significativos foi o da indústria da manufatura, moldando, assim, muitas outras indústrias [1]. Estes avanços tecnológicos manifestam-se nas primeiras utilizações de ferramentas como semicondutores, computadores e robótica, bem como na crescente implementação de automação em linhas de produção. Esta tendência, mais acentuada após 1950, viria a ser reconhecida como o início das práticas associadas à terceira revolução industrial.

A evolução da internet, aliada à evolução de sensores compactos e robustos, ao avanço acelerado da capacidade de processamento e à emergência de sistemas integrados, catalisou a transformação das fábricas em redes interconectadas de dispositivos. Estes não só interagem entre si, como também comunicam com seres humanos, no conceito conhecido como *Internet of Things* (IoT), inauguraram uma nova era industrial. Esta fase foi cunhada pelos professores Erik Brynjolfsson e Andrew McAfee, do Instituto de Tecnologia de Massachusetts, como 'segunda idade da máquina' e é atualmente designada como indústria 4.0 [2].

Os conceitos "Indústria 4.0", "*smart factory*", "fábrica inteligente" e "fábrica do futuro" delineiam uma projeção do futuro da produção industrial [3]. Neste panorama, as fábricas são visualizadas como entidades mais inteligentes, adaptáveis, dinâmicas e ágeis. Uma "s*mart factory*" não se refere apenas a uma fábrica dotada de inteligência, mas também à capacidade de produzir artigos inteligentes, utilizando equipamentos avançados e integrados em cadeias de abastecimento igualmente sofisticadas.

Considerando a larga escala de problemas industriais advindas desta interligação, o sucesso e estabilidade destes processos decorrem do eficiente poder preditivo de

algoritmos de *Machine Learning* (ML). Deste modo, investigadores e empresas procuram cada vez mais novas maneiras de utilizar esta ferramenta.

O desenvolvimento da aplicação de ML e IoT tornaram possível a redução de custos em indústrias, de modo que garante a qualidade do produto e a segurança dos funcionários. Na indústria onde humanos desempenham funções, para é inevitável que ocorram erros. No entanto, com a integração de ML e IoT, assiste-se à transformação destas indústrias em sistemas totalmente automatizados. Atualmente, desde a manutenção de máquinas até ao controlo de qualidade pós-produção, tudo pode ser gerido através destas tecnologias. Em pleno processo produtivo, é viável monitorizar o estado de máquinas e produtos recorrendo a sensores IoT. Paralelamente, a deteção de anomalias, tanto em produtos como em máquinas, é facilitada pelo poder do ML [4].

Apesar dos avanços significativos proporcionados por esta integração, ainda há desafios a ser superados a fim de alcançar pleno potencial da Indústria 4.0 e iniciar a transição para revoluções industriais seguintes. A precisão e fiabilidade destes novos sistemas que fazem uso de ML ainda tem que ser estudada, devido a variabilidades intrínsecas destes sistemas. Além disso, a integração dessas plataformas com as linhas de produção existentes deve ser refinada para garantir uma operação contínua e eficiente.

1.2 Problema em estudo

Em 2015, um estudo realizado pela *The Food and Agriculture Organization* (FAO) estimou que 30% dos alimentos produzidos para consumo humano a nível mundial são perdidos ou desperdiçados algures ao longo da cadeia de abastecimento alimentar [5]. Até 2050, a população mundial deverá alcançar a marca de 9,1 mil milhões de pessoas, aumentando a procura de alimentos em 70% [6].

A perda e o desperdício de alimentos têm impactos negativos significativos na segurança alimentar, na economia e no ambiente. O desenvolvimento de soluções eficientes para reduzir a perda e o desperdício de alimentos passa, por exemplo, pelo reconhecimento da integridade e qualidade do produto hortofrutícola entre as diferentes fases da cadeia de abastecimento alimentar.

A qualidade das frutas e legumes está relacionada com seu grau de maturação. Este grau pode ser determinado com a combinação de atributos físicos e químicos, que determinam a aceitabilidade pelos consumidores [7]. Existem cinco diferentes atributos

que definem a qualidade dos hortofrutícolas: qualidade visual (tamanho, forma, cor e defeitos), textura, sabor (gosto e aroma), valores nutricionais e segurança [8].

Em geral, a qualidade das hortofrutícolas é avaliada por determinação sensorial e subjetiva, através de escalas de pontuação, cujos níveis de qualidade são caracterizados por uma breve descrição e imagens exemplificativas do produto em causa [9]. Além disso, são utilizados métodos destrutivos convencionais para medir os atributos químicos e físicos dos frutos e produtos hortícolas para apoiar a avaliação sensorial.

As técnicas analíticas e destrutivas são amplamente utilizadas, embora sejam caras e demoradas, e tendo um impacto negativo no ambiente e no produto. Essas técnicas exigem equipamentos sofisticados e uma preparação cuidadosa das amostras [10].

A visão computacional representa a entrada de uma tecnologia não destrutiva ideal para a avaliação da qualidade de frutos e produtos hortícolas. Esta tecnologia pode ser implementada para aplicações de linhas industriais, representando uma opção fiável e adequada para este ambiente. Outra aplicação é em robôs autónomos utilizados no setor agroindustrial, a visão computacional pode permitir tanto a locomoção do robô como tarefas de manipulação, sendo uma alternativa viável e com uma manipulação delicada.

Assim, o avanço desta tecnologia oferece uma alternativa aos métodos tradicionais de avaliação de qualidade em hortofrutícolas. Além disso, a implementação de um sistema de visão computacional pode reduzir custos operacionais com uma instalação mais simples e de fácil integração em ambientes específicos.

1.3 Objetivos

Esta dissertação tem como objetivo desenvolver um sistema de visão computacional que possa substituir ou trabalhar em conjunto com sensores tradicionais, ou seja, ser um método que possa ser implementado com facilidade que utilize a visão computacional como principal ferramenta para obtenção de dados do nível de qualidade de frutas e legumes. Esta dissertação também terá como foco a criação deste sistema utilizando equipamentos com baixo custo operacional. O foco desta dissertação será então a implementação deste sistema num ambiente em que um braço robótico colaborativo é o principal atuador para manuseamento dos produtos, em particular, de frutas e legumes mais sensíveis ao dano mecânico decorrente do seu manuseamento.

Esta dissertação usa linhas de produção tradicionais como ambiente de estudo. O projeto desenvolve-se para ser uma implementação prática na Cerfundão, empresa agroalimentar localizada no Fundão, Cova da Beira, Portugal. Esta empresa trabalha com diversas frutas, mas tem um foco principal na cereja e no pêssego, possuindo diferentes equipamentos para cada tipo de fruta. Uma vez que a linha de seleção de pêssego é a menos automatizada, foi então foco deste projeto a sua futura implementação numa linha semelhante, podendo atuar em frutos de tamanho semelhante a um pêssego, no caso laranjas e tomates. Esses dois frutos foram escolhidos pela facilidade de serem obtidos ao longo de todo ano.

1.4 Estrutura da dissertação

Esta dissertação tem como objetivo principal a implementação de um sistema de visão computacional para o controlo de qualidade de frutas e vegetais e oferece uma visão abrangente dos resultados obtidos pelo sistema criado. A estrutura da dissertação segue o seguinte esquema:

1. Introdução: Nesta secção é apresentado o contexto geral do trabalho desenvolvido, destacando a importância do estudo de novas tecnologias no mercado alimentar. Discute-se a evolução da integração de tecnologias recentes e os objetivos desta dissertação.
2. Estado da Arte: Nesta secção é analisada a influência do surgimento e aplicação de novas tecnologias, demonstrando as aplicações das principais tecnologias na formação da atual revolução industrial no setor alimentar/agrícola. O foco principal deste capítulo é contextualizar a visão computacional e os seus principais campos de atuação. Além disso, são abordadas as pesquisas que serviram de base e incentivo para este estudo.
3. Materiais e Métodos: Neste capítulo são inicialmente caracterizados os principais materiais utilizados neste projeto, tanto na perspetiva de *hardware* como de *software*. Em seguida, são discutidas as estratégias de utilização e os métodos implementados no sistema final, que também é descrito detalhadamente neste capítulo.
4. Resultados: Neste capítulo são apresentados os resultados obtidos por cada modelo treinado, tanto no campo da eficiência do modelo em si, como os resultados de efetividade geral do sistema.

5. Análise e Discussão de Resultados: Neste capítulo, os resultados são analisados e discutidos em detalhe, relacionando os dados de efetividade numérica do modelo com os resultados reais obtidos no sistema.
6. Conclusão: As conclusões gerais são apresentadas, resumindo as principais contribuições dos resultados obtidos por esta dissertação. São apresentadas sugestões de trabalhos futuros que possam expandir e aprofundar o conhecimento nesta área, como também sugestões de estudos futuros para melhorar o sistema iniciado neste projeto.

Capítulo 2

Estado da arte

Neste capítulo, é apresentado um panorama detalhado do cenário atual no campo de estudo em foco, sublinhando os temas principais que fundamentam esta dissertação. É destacada a relevância e a atualidade destes temas. Além disso, é feita referência aos estudos que serviram de inspiração para este trabalho, detalhando os elementos-chave que caracterizam cada tema abordado.

2.1 Influência das revoluções industriais na agricultura e indústria alimentar

A segurança alimentar é um conceito multidimensional que mitiga a fome assegurando um fornecimento sustentável e nutritivo de alimentos. Esta é composta por um modelo de quatro pilares, ilustrado na Figura 1 [11]. As evoluções tecnológicas na agricultura têm sido indispensáveis para a este propósito.

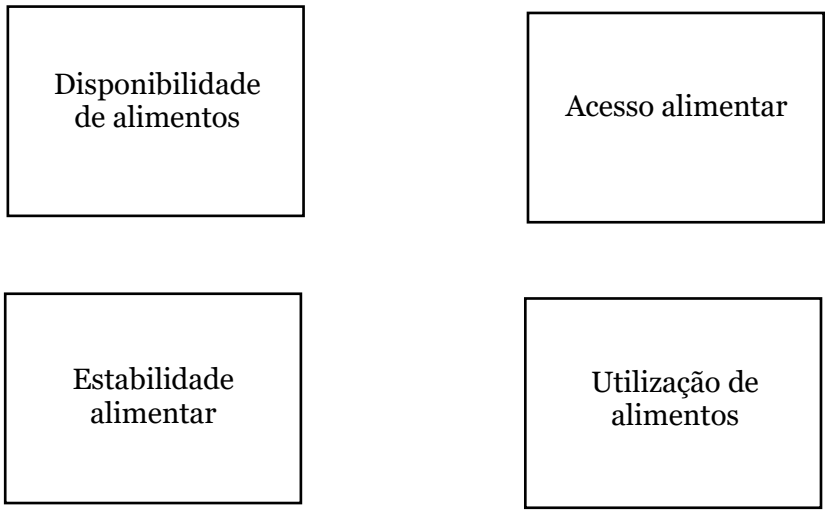


Figura 1 Modelo de quatro pilares de segurança alimentar ([11]).

A partir de um ponto de vista histórico e tecnológico, até a década de 1950, a “Agricultura 1.0” era descrita como intensiva, mas de baixo rendimento. Na década de 1960, surgiu a "Agricultura 2.0" como a primeira geração de genética e hibridização, o uso de energia fóssil (fertilizantes, agroquímicos, maquinaria) e por consequência, aumentos de produtividade. A “Agricultura 3.0” é descrita pelo desenvolvimento das aplicações do conhecimento em todas as áreas, e particularmente, na busca de um equilíbrio entre produtividade e melhor desempenho ambiental [12]. Foi neste período que se deu início

à utilização de energias renováveis verdes, como energia fotovoltaica, hídrica, e energia eólica, e ao uso de tecnologias da informação para agricultura de precisão [13], através da monitorização de rendimento, aplicações de taxa variável e sistemas de orientação agrícola. A quarta revolução industrial (Indústria 4.0), ainda em curso, caracteriza-se pela fusão de tecnologias como a Internet das Coisas (IoT), robótica, *big data*, inteligência artificial (IA) e tecnologia *blockchain*. Atualmente, os processos de produção industrial e as cadeias de abastecimento tornaram-se mais autónomos e inteligentes [14]. Correspondentemente, a integração da Indústria 4.0 na agricultura está a transformar a agricultura industrial, fenómeno chamado de Agricultura 4.0 [15]. Neste cenário, uma agricultura industrial sustentável e inteligente seria alcançada através da recolha, processamento e análise em tempo real e de forma detalhada de dados espaço-temporais em amplos aspetos da indústria agrícola, desde a produção de alimentos, processamento, distribuição até à experiência do consumidor [16]. A evolução da agricultura mediante esta revolução industrial está representada na Figura 2.

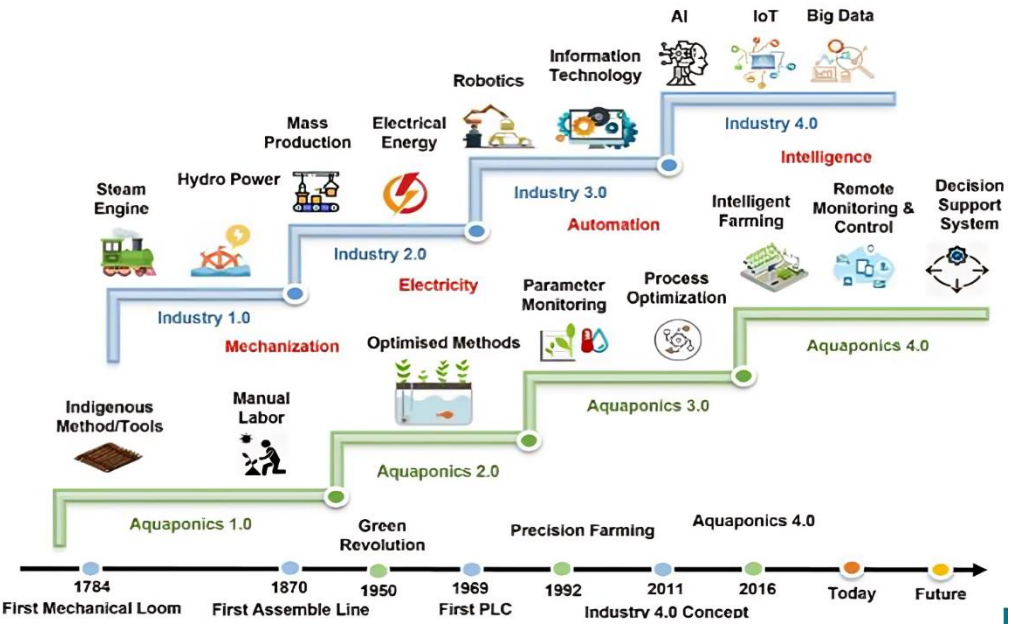


Figura 2 Desenvolvimento das revoluções industriais e das revoluções agrícolas [17].

2.1.1 Internet das coisas na indústria alimentar

A Internet das coisas (*Internet of Things* - IoT) é uma rede que liga dispositivos a pessoas, em qualquer lugar e a qualquer momento. Aos sistemas que se conseguem conectar e comunicar, ou seja, capacidade de enviar e receber dados com outros dispositivos, é denominada de "Internet das Coisas" [17]. A tecnologia IoT permite a monitorização e controlo remotos de objetos físicos através de uma rede [18]. A IoT possui capacidades de transmissão e receção de dados por meio de comunicação. Esta

tecnologia pode fornecer informações de dados em tempo real ao utilizador, sendo útil para o monitorização e reparação de problemas não esperados.

A IoT é um conjunto de "coisas" interconectadas e está principalmente preocupada com tecnologias de rede, comunicação, deteção e processamento de dados [19]. Entre as práticas mais comuns da IoT, encontram-se as casas inteligentes [20], sistemas de reabilitação inteligentes [21], cuidados de saúde domiciliários [22], gestão inteligente da cadeias de abastecimento [23], como também no setor agrícola [24].

Na indústria alimentar, há um destaque para monitorização e controlo dos equipamentos de processamento empregues no setor, como fornos, misturadores, secadores e transportadores, em tempo real [25], esta é uma prática crucial para garantir desempenho, produtividade e segurança de alta qualidade. Com a abordagem da IoT, os operadores podem ajustar as configurações e características do sistema sem intervenção física, graças às funcionalidades do controlo remoto. Outra grande vantagem da IoT aplicada ao setor alimentar é o auxílio na manutenção preditiva. Através da monitorização de equipamentos de processamento de alimentos, problemas potenciais são detetados antes que causem uma avaria. Isso ajuda a prevenir longos períodos de inatividade na linha de produção e reduz os custos de manutenção [26].

Para controlo de qualidade, sensores IoT podem ser utilizados para realizar análises químicas de amostras de alimentos, os quais podem detetar a presença de substâncias nocivas como pesticidas ou metais pesados [27]. Estes sensores podem ser instalados em vários pontos da cadeia de abastecimento, incluindo instalações de produção, armazéns e lojas de retalho. Dispositivos IoT podem ser equipados com câmaras para realizar análises de imagem em amostras de alimentos, detetando quaisquer alterações físicas ou anormalidades que possam indicar adulteração [28]. Em geral, a tecnologia IoT pode detetar a adulteração de alimentos, contribuindo assim para a segurança e integridade da cadeia de abastecimento de alimentos.

2.1.2 Robótica na indústria alimentar

As exigências de alta produtividade em aplicações industriais obrigaram a colocação de robôs para automatizar várias tarefas [29]. Hoje, os robôs são considerados como parte integrante das indústrias devido ao seu papel na melhoria da precisão, repetibilidade, fiabilidade, exatidão e eficiência dos processos [30]. O uso de robôs ajuda a eliminar lesões, melhorar a frequência e qualidade da produção, reduzir os custos diretos de mão-de-obra e aumenta a segurança [31], enquadrando-se na 5ª revolução industrial, centrada no Homem, que nos encontramos a iniciar. Na agricultura e na indústria

alimentar, os robôs estão a ser usados em todas as tarefas, ou seja, desde a sementeira, pulverização de fertilizantes, herbicidas e pesticidas, irrigação e colheita até ao corte, processamento e embalagem de produtos alimentares [32].

O controlo de manipuladores industriais é importante para realizar tarefas que exigem alta precisão, confiabilidade e repetibilidade, mitigando os efeitos de distúrbios [33]. Abordagens de controlo, geralmente baseadas em leis de controlo lineares como Proporcional Integral Derivativo (PID) [34], Regulador Quadrático Linear (LQR) [35] e controlo Linear Quadrático Gaussiano (LQG) [36] foram as principais estratégias de controlo até meados do início da década de 2010 [37]. No entanto, a tendência recente é empregar estratégias de controlo modernas e não lineares em aplicações do mundo real [34]. A Figura 3 ilustra o desempenho superior de um Controlo por Modo Deslizante (SMC) sobre uma estratégia linear [38].

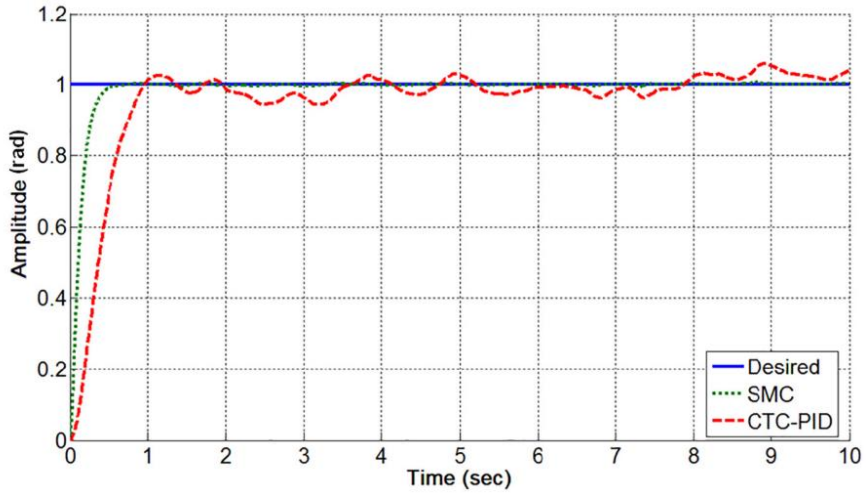


Figura 3 Desempenho comparativo de aderência na presença de perturbações: Respostas ao degrau do SMC e CTC [39]

Ao projetar controlo destreza, é necessário garantir que sejam permitidos os suficientes Graus de Liberdade (*Degree of Freedom* - DOF) para o efetuador final se mova em todos os eixos desejados [40].

Os robôs na indústria alimentar são principalmente classificados conforme o uso em várias tarefas. Os diversos tipos de robôs que podem se encontrar nesta indústria estão apresentados na Tabela 1.

Tabela 1 Robôs comuns utilizados na indústria alimentar [40]

Tipos de robôs	Descrição	Função	Exemplo
Robôs pórticos	Montados com três eixos lineares que abrangem uma área cúbica de manuseio, com cinemática robótica situada acima da montagem.	Cobrir uma grande área de trabalho com movimento preciso.	Portais Liebherr
Robôs articulados	Múltiplos braços articulados com juntas que podem ser equipados com garras, oferecem flexibilidade e até seis graus de liberdade.	Executar tarefas complexas como soldagem, pintura e montagem.	KUKA KR AGILUS
SCARAs (Selective Compliance Assembly Robot Arms)	Braços articulados horizontais, trabalham em série.	Movimento horizontal para montagem e manuseio de materiais.	Fanuc SR-3iA
Robôs delta	Estrutura similar a uma aranha com atuadores estacionários, permitindo alta velocidade e aceleração	Picking e packing em alta velocidade, especialmente em alimentos e farmacêuticos.	ABB FlexPicker e RacerPack

2.1.3 *Big Data* e *blockchain* na indústria alimentar

Big Data, de forma geral, refere-se à aquisição de grandes quantidades de dados e à sua transformação em pedaços de informação geríveis, o que ajuda a alcançar estatísticas abrangentes e relevantes sobre um tópico [41].

Na essência, *blockchain* trata-se de uma nova arquitetura descentralizada e um paradigma de computação distribuída que utiliza uma estrutura de bloco de cadeia de criptografia assimétrica para verificar e armazenar dados. Usa algoritmo de consenso de nó distribuído para gerar e atualizar dados, e utiliza contratos inteligentes para programar e operar dados [42].

A quantidade de dados gerados por cada tipo de indústria agroalimentar é imensa [43]. A gestão tradicional da segurança da informação depende principalmente de medidas como o fortalecimento do nível de defesa das bases de dados, o reforço da gestão da

confidencialidade [44]. Na cadeia de abastecimento alimentar, ferramentas digitais de rastreio, sensores remotos e dispositivos impulsionados pela IA criam uma cadeia de abastecimento segura do produtor ao consumido [45] .

A indústria agroalimentar possui ligações complexas, e a cadeia de abastecimento alimentar inclui sozinha cinco elos típicos: plantação, transporte, armazenamento, processamento e venda [46].

O contrato inteligente de *blockchain* é interpretado como uma nova tecnologia que pode propagar, verificar, negociar automaticamente, executar e fazer cumprir os termos de um acordo de forma informativa num ambiente de *blockchain* [43]. Comparados com contratos tradicionais, os contratos inteligentes de *blockchain* apresentam baixo custo, alto grau de automação, alta eficiência e alta segurança [47]. Os contratos inteligentes de *blockchain* possuem características como disparo condicional, execução automática, baixo custo e descentralização. Em estudos anteriores, foi explorada a aplicação de contratos inteligentes de blockchain na gestão das indústrias agroalimentares [48]. Neste âmbito, realizaram deteção automática de qualidade e alerta precoce nas cadeias de abastecimento agroalimentar [49], captura automática do estado e ciclo do produto [50], rastreio automático de tendências do produto [51] e tomada de decisões automáticas de eventos relacionados [52], automatizando tarefas regulamentadas tediosas. Além disso, os contratos inteligentes de *blockchain* são utilizados para regular e monitorizar transações na indústria agroalimentar, e mostram processos regulamentares detalhados aos *stakeholders*, melhorando assim a transparência do processo [53].

2.1.4 Inteligência Artificial na indústria alimentar

A IA surgiu como cálculo de implementação e tecnologias de grandes dados para desbloquear, quantificar e interpretar padrões de dados complicados. Esta tecnologia tem sido utilizada como resposta as adversidades técnicas complexas e ganhou popularidade nos últimos anos [54]. A IA encontra-se aplicada numa vasta gama de sistemas de alimentação e agricultura, desde a produção [55] ,deteção de doenças [56] até embalagem e distribuição ao consumidor [57].

As técnicas de IA são benéficas como elementos de sistemas integrados ou como uma abordagem alternativa para substituir técnicas tradicionais. A IA pode lidar com dados com variações aleatórias e/ou ausentes, pois pode aprender com exemplos, é capaz de

lidar com equações não lineares e é capaz de realizar previsões e generalizações de alta velocidade uma vez treinada [58].

Uma variedade de métodos baseados em IA emergiu em diferentes campos de estudo científico e melhoria tecnológica, incluindo o estudo da qualidade dos alimentos [59]. A sua eficácia na deteção de danos e anormalidades em produtos agrícolas [60] demostra que sua utilização é uma tendência dentro do controlo de qualidade desta indústria.

Recentemente [61], apresentou uma técnica para detetar automaticamente em tempo real imperfeições de múltiplos tipos em batatas através de deteção de imagens multiespectral integrada com CNN, cujo exemplo se encontra ilustrado na Figura 4.

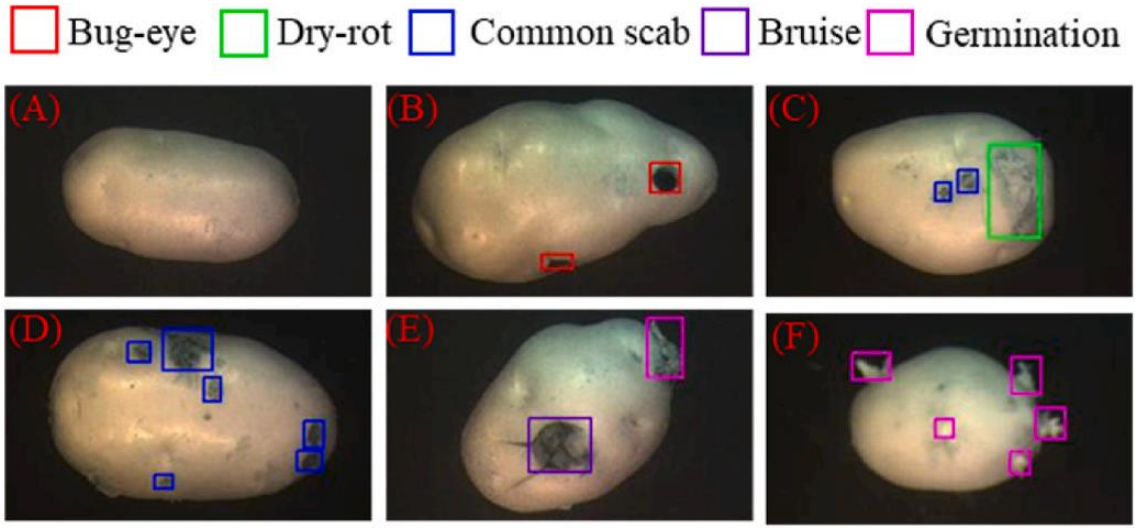


Figura 4 Deteção de defeitos em batatas utilizando visão computacional [61].

Nenhum algoritmo único tem a capacidade de resolver todos os problemas. Selecionar o algoritmo de aprendizagem ótimo para um problema específico é crítico para o desempenho do modelo [62]. A seleção de algoritmos tem sido tradicionalmente uma prática de tentativa e erro, no entanto, várias investigações adotaram uma estratégia comparativa, na qual múltiplos algoritmos são utilizados para um problema de classificação e o melhor algoritmo é escolhido através de um compromisso entre características de desempenho [63].

2.2 Visão Computacional

A visão computacional (CV) refere-se ao campo de investigação que visa desenvolver métodos que permitam aos computadores interpretar e compreender o conteúdo de

imagens e vídeos [64]. Trata-se de uma área multidisciplinar que se situa como uma subárea da inteligência artificial e ML (Figura 5).

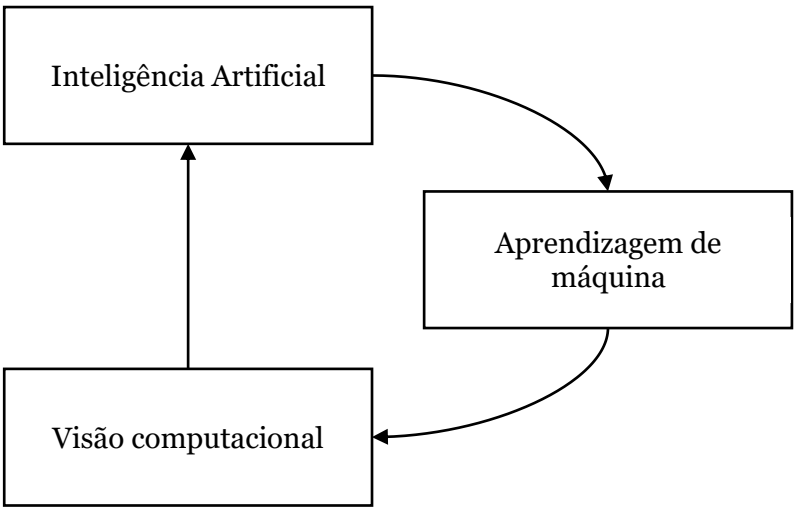


Figura 5 Visão Geral da Relação entre Inteligência Artificial e Visão Computacional (Adaptado de [65]).

A CV mistura técnicas de processamento de imagem com reconhecimento de padrões [66] O resultado da CV é a interpretação ou compreensão da imagem. A evolução deste campo tem sido pautada na adaptação das habilidades da visão humana para a extração de informações e abrange um extenso campo que vai desde a captação de dados em bruto, até à extração de padrões de imagem e interpretação de informações [67].

Distintamente da Computação Gráfica, a visão computacional foca-se na obtenção de informações a partir de imagens. O progresso nesta área está intrinsecamente ligado à evolução da tecnologia computacional, seja na melhoria da qualidade da imagem ou no seu reconhecimento. É importante notar que há uma intersecção entre a visão computacional e o processamento de imagem em relação a técnicas fundamentais, levando alguns autores a usarem os termos de forma análoga [68]. O principal propósito da CV é desenvolver modelos para extrair dados e informações de imagem. Por outro lado, o Processamento de imagem concentra-se na realização de transformações computacionais nas imagens, como melhoria da nitidez, ajuste de contraste e outras alterações.

O significado de CV também se assemelha e, em certas ocasiões, sobrepõe-se ao de Interação Humano-Computador (*Human–Computer Interaction* - HCI) [69]. A HCI foca o design completo, interface e todos os aspetos vinculados à interação entre seres humanos e máquinas. Posteriormente, HCI estabeleceu-se como uma disciplina distinta - um campo da ciência interdisciplinar - que aborda as interconexões mediadas pela

tecnologia entre humano e computador, englobando aspetos humanos. Tal como se pode observar no esquema da Figura 6, a visão computacional e a visão humana são similares, com a meta de interpretar dados espaciais, ou seja, dados mapeados em múltiplas dimensões [70]. Contudo, não se deve esperar que a visão computacional imite exatamente a capacidade visual humana [71].

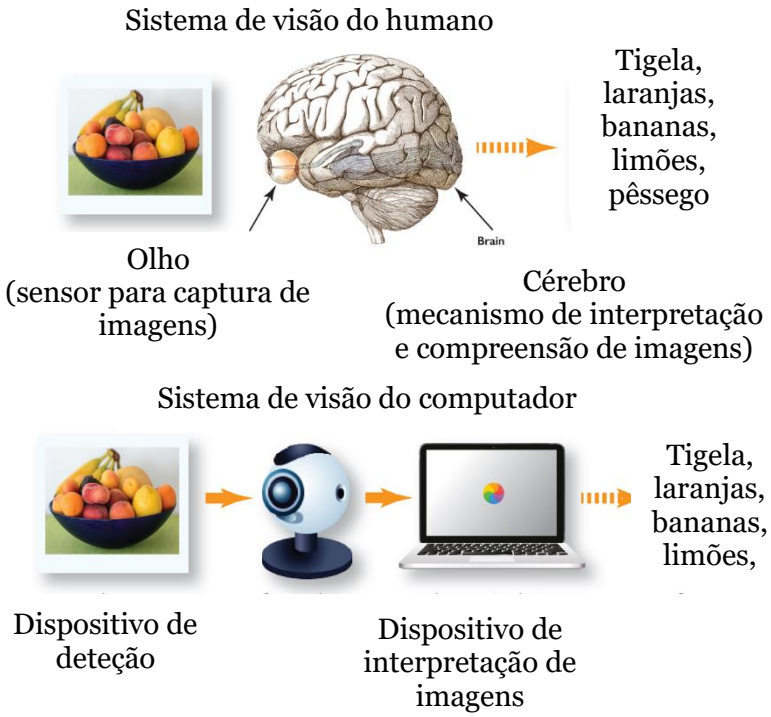


Figura 6 Comparação entre visão humana e visão computacional (Adaptado de [72]).

O sistema de CV apresenta desempenho e funcionalidades limitadas quando comparado à visão humana. Apesar de muitos investigadores proporem uma vasta gama de técnicas de visão computacional para imitar a capacidade visual humana, frequentemente, o desempenho do sistema de visão computacional encontra-se restrito [73]. Uma das principais barreiras técnicas é a sensibilidade aos parâmetros, robustez do algoritmo e precisão dos resultados obtidos. Estas nuances aumentam a complexidade na avaliação de desempenho desses sistemas. Em geral, a avaliação de desempenho exige a medição de comportamentos fundamentais de um algoritmo com o intuito de determinar a sua precisão, robustez ou capacidade de adaptação, tudo isso visando gerenciar e monitorizar eficazmente o rendimento do sistema.

2.2.2 Evolução da Visão Computacional

A visão computacional opera mediante a aplicação de algoritmos e sensores óticos para simular a capacidade de visão humana, com o propósito de extrair automaticamente

informações relevantes de um objeto [74]. Em contraste com os métodos tradicionais, que são demorados e requerem análises laboratoriais avançadas, a visão computacional evoluiu para se tornar uma vertente da inteligência artificial, simulando assim a perceção visual humana.

Estes algoritmos são desenvolvidos com recurso ao *Deep Learning* (DL), uma subárea preponderante da IA. O DL, com as suas estruturas de rede diversificadas, tem sido amplamente investigado e aplicado. As características adquiridas podem ser detetadas pelo DL de maneira automática e eficaz.

Nas fases iniciais do desenvolvimento da CV, a abordagem baseada em DL enfrentou desafios devido às limitações da memória do computador, CPU e GPU. A maioria dos trabalhos de investigação focava-se, assim, na investigação da aplicação ML em CV [75]. Entretanto, muitos métodos para CV foram propostos, tais como *K-means*, Classificador Naive Bayes, Árvore de Decisão, *Boosting, Random Forest*, Classificador Haar, *Expectation–Maximization*, *K-Nearest Neighbor* e *Support Vector Machine* [75]. Com base no algoritmo Adaboost, Viola e Jones [76] utilizaram a característica *wavelet* tipo Haar e o método gráfico integral para deteção facial. Não foram os primeiros a propor características *wavelet*, mas desenharam características de deteção facial mais úteis e encadearam o classificador forte treinado pelo Adaboost. O algoritmo proposto é denominado detetor Viola–Jones [77]. Mais tarde, Lienhart e Maydt (2002) ampliaram este detetor com características *Haar-like* rodadas, culminando no classificador Haar que o OpenCV atualmente possui, exemplificado na Figura 7.

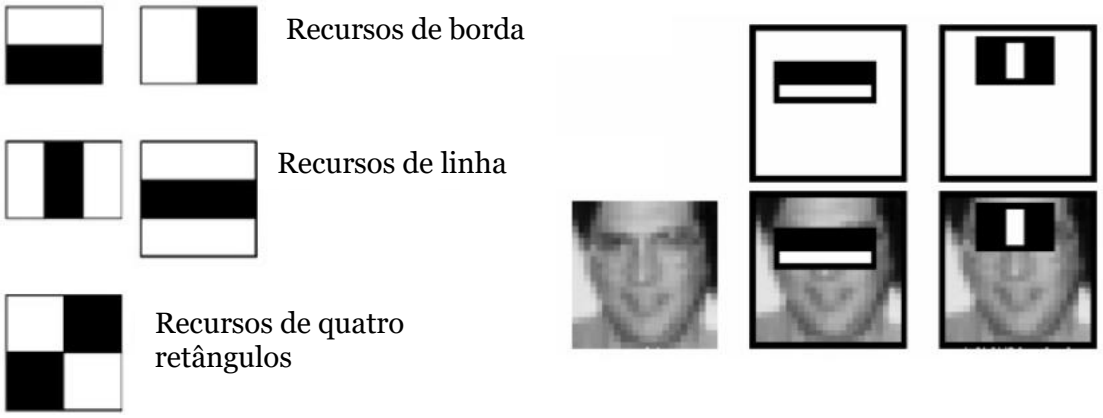


Figura 7 Exemplo de funcionamento do classificador Haar (Adaptado de [78]).

Os desenvolvimentos em DL nas últimas décadas têm sido bastante rápidos e podem ser divididos de forma geral em dez categorias, em termos de algoritmo e arquitetura: Redes

Neuronais Convolusionais (*Convolutional Neural Network* - CNNs), Redes de Memória de Curto e Longo Prazo (*Long Short-Term Memory* - LSTMs), Redes Neuronais Recorrentes (*Recurrent Neural Networks* - RNNs), Redes Adversárias Generativas (*Generative Adversarial Network* - GANs), Redes de Função de Base Radial (*Radial Basis Function Network* - RBFNs), Perceptrons Multicamadas (*Multilayer Perceptron* - MLPs), Mapas Auto-Organizáveis (*Self-Organizing Map* - SOMs), Redes de Crença Profunda (*Deep Belief Network* - DBNs), Máquinas de Boltzmann Restritas (*Restricted Boltzmann Machine* - RBMs) e Autoencoders [75]. Ao comparar CNN, RBM, Autoencoder e Codificação Esparsa, através da literatura e o respetivo desempenho em diferentes tarefas de CV, incluindo classificação de imagens, deteção de objetos, recuperação de imagens, segmentação semântica e estimativa de pose humana, concluíram que a CNN era a arquitetura mais adequada para CV [79]. No entanto, devido às limitações de precisão e tamanhos de modelos da época, havia vários desafios na aplicação prática. Existia uma falta de compreensão clara sobre qual arquitetura se deveria destacar em relação às demais. Adicionalmente, enfrentava-se o desafio de treinar com um conjunto de dados restrito e havia uma complexidade considerável em desenvolver aplicações que operassem em tempo real. Para além disso, sentia-se a urgência por modelos com maior capacidade e robustez. Desde a notável atuação na competição ImageNet, as CNNs tornaram-se as abordagens de DL mais proeminentes [79]. Uma das áreas mais fundamentais para aplicações em CV é a classificação de imagens, esquematizada na Figura 9.

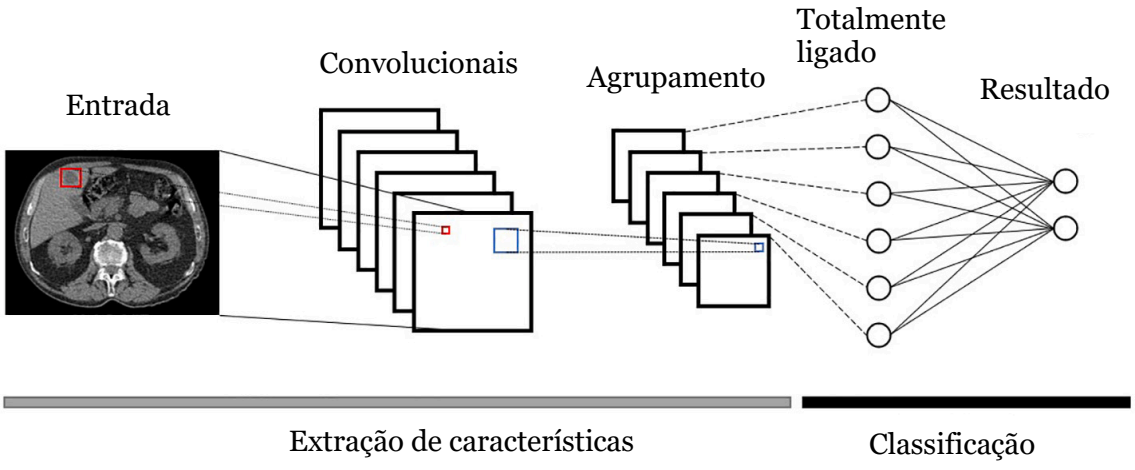


Figura 8 Arquitetura CNN para classificação de imagens [80].

Uma CNN é composta por camadas convolucionais, camadas de agrupamento e camadas totalmente conectadas. Nas camadas convolucionais, diferentes núcleos são utilizados para convolver a imagem completa e os mapas intermediários de características,

resultando em diversos mapas de características. As camadas de agrupamento têm a função de reduzir as dimensões desses mapas e os parâmetros da rede. Quanto às camadas totalmente conectadas, geralmente situam-se no final da arquitetura CNN, funcionando como um classificador. Também é possível a saída ser redirecionada para subsequentes Redes Neuronais Profundas (*Deep Neural Network* - DNN) para se obter modelos derivados [75].

2.2.3 Classificador de imagem

A classificação de imagens visa atribuir um rótulo pré-definido a uma entrada. Tal como se pode observar na Figura 9, existem múltiplas arquiteturas de redes que podem ser utilizadas para classificação de imagens, destacando-se: AlexNet [81], VGGNet [82], GoogleLeNet & Inception [83], ResNet [84], DenseNET [85], MobileNet [86], EfficientNet [87],RegNet [88], entre outras.

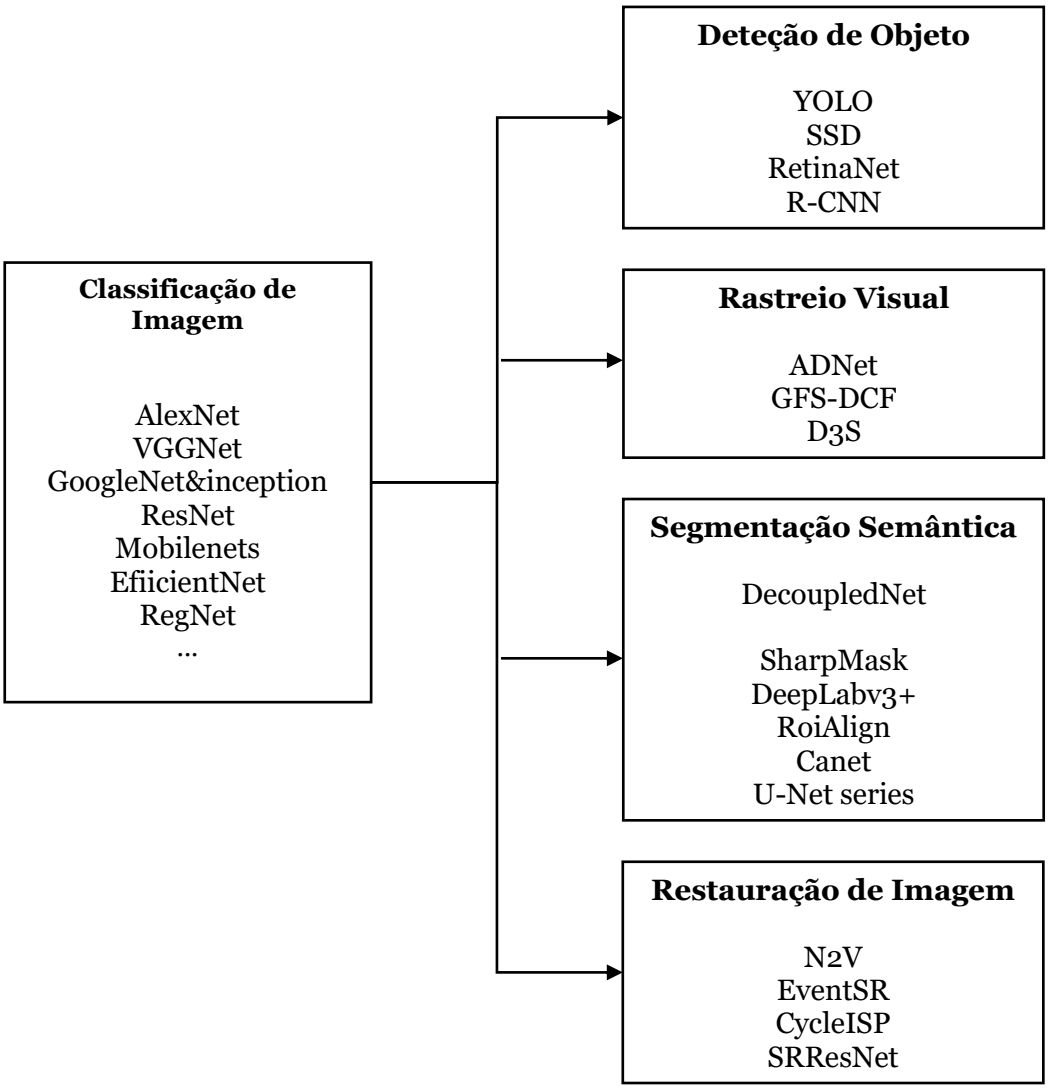


Figura 9 Modelos básicos da CNN e modelos derivados (Adaptado de [75]).

Cada um dos modelos referidos constituiu um marco e distinção no domínio da CV. O AlexNet, introduzido em 2012, distinguiu-se como pioneiro pela sua vitória no desafio ImageNet, consolidando-se como um modelo de referência e uma fonte de inspiração para os desenvolvimentos subsequentes nesta área. Estes modelos diversificados procuram especializar-se em aspetos específicos da CV. Por exemplo, o AlexNet e o VGGNet focam-se essencialmente na profundidade e na eficiência das operações convolucionais. Em contrapartida, o GoogleNet representou uma inovação ao introduzir uma arquitetura enriquecida com os módulos Inception, desenhados para otimizar a alocação dos recursos computacionais. Já o ResNet marcou uma revolução no treino de redes de elevada profundidade através da implementação de conexões residuais. O DenseNet destacou-se pela sua capacidade de interligar todas as camadas diretamente, potenciando a propagação de características e diminuindo o número de parâmetros necessários. Por outro lado, o MobileNet inaugurou uma gama de modelos eficientes, equilibrando latência e precisão através de hiperparâmetros globais simplificados. O EfficientNet, por sua vez, alcançou um equilíbrio ótimo entre profundidade, largura e resolução da rede, visando uma performance superior. Finalmente, o RegNet combinou as vantagens do design manual com a busca automatizada de arquiteturas de rede, superando modelos antecessores em termos de velocidade de processamento em GPU.

2.2.3.1 Deteção de objetos

A deteção de objetos visa localizar objetos numa imagem e fornecer etiquetas de classificação para esses objetos [89]. O principal objetivo é localizar e classificar de maneira precisa os objetos numa imagem, incluindo segmentações semânticas [90]. A deteção de objetos combina a localização e a classificação de objetos numa única tarefa, sendo precisão e o desempenho em tempo real os principais indicadores da eficiência do algoritmo [91]. Em cenários mais complexos, onde vários objetos devem ser processados em tempo real, a extração e reconhecimento automático de objetos são particularmente importantes. Deste modo, o algoritmo necessita de processar uma grande quantidade de dados rapidamente [91].

Antes dos métodos de DL serem amplamente utilizados, os algoritmos de deteção de objetos baseavam-se principalmente em técnicas geométricas, focando-se na disposição espacial de objetos específicos, e os modelos eram relativamente simples. Baseados no algoritmo tradicional de deteção de objetos com características desenhadas manualmente, os passos e processos de deteção incluem principalmente pré-processamento, seleção de região, extração de características e processos de classificação de características. Com o desenvolvimento do DL, os recursos de computação evoluíram.

O apoio da tecnologia tem permitido que os modelos de deteção de objetos baseados em DL registem avanços importantes em termos de precisão e eficiência. Neste contexto, emergiram trabalhos de investigação clássicos como o *Single Shot Detector (*SSD*)* [92], *You Only Look Once (*YOLO*)* [80] e *The Detection Transformer* (DETR) [93].

Diferente do método tradicional, o método baseado em DL evita o desenho manual e pode aprender características mais profundas com maior poder de diferenciar. Ao mesmo tempo, as abordagens baseadas em aprendizagem profunda unificam a extração de características e a aprendizagem do classificador num único quadro, possibilitando a aprendizagem de ponta a ponta [91].

Os algoritmos de deteção de objetos com DL mais avançados dividem-se em duas categorias: algoritmos de deteção de objetos de uma e duas fases. Nos algoritmos de deteção de objetos de duas fases, várias regiões de interesse de características são geradas na primeira fase; na segunda fase, os vetores de características das regiões geradas na primeira fase são codificados por uma rede neuronal para prever a classe do objeto e localizá-lo. O algoritmo de deteção de objetos de uma fase considera todas as localizações na imagem como regiões de interesse e tenta classificar cada região para distinguir o fundo do objeto [91].

Os métodos de deteção de objetos de uma fase, como YOLO e SSD, permitem a deteção de imagens de forma mais rápida que os detetores de duas fases. YOLO transforma a deteção num problema de regressão, usando toda a imagem como entrada para a rede e regredindo diretamente a localização e categoria na camada de saída. A estrutura desta rede é apresentada na Figura 10. As versões subsequentes do YOLO, como YOLOv2 [94], YOLOv3 [95], YOLOv4 [96], YOLOv5 [97], YOLOX [98] e YOLOv6 [99], introduzem melhorias contínuas, incluindo técnicas de normalização em lote, classificação de alta resolução, ancoras, redes separadas para classificação e regressão, e estratégias de autoaprendizagem. O YOLOv7 [100] foca em otimizar o processo de treino, enquanto o YOLOv8 [101] , além da deteção de objetos, suporta segmentação de instâncias e utiliza a arquitetura Darknet-53 para uma extração de características mais eficaz, combinando elementos de versões anteriores para alcançar velocidades mais rápidas e maior precisão.

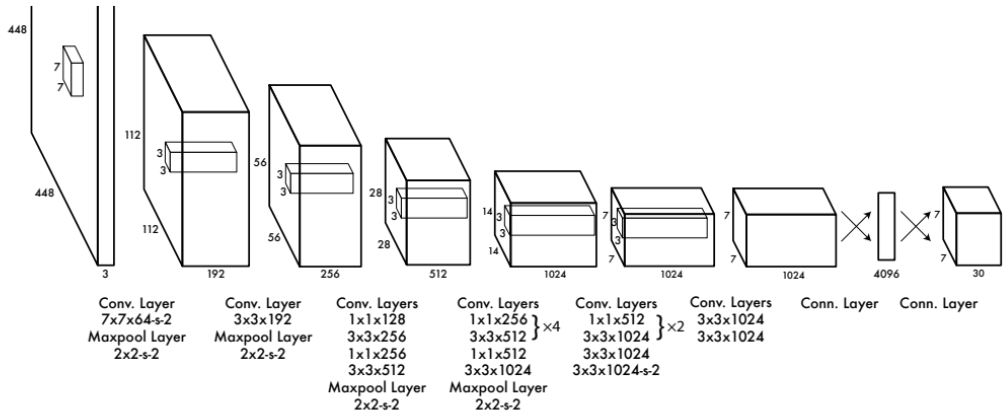


Figura 10 Estrutura YOLO [102]

O SSD utiliza pequenos núcleos convolucionais para prever pontuações de categoria e deslocamentos de caixas delimitadoras, gerando mapas de características multiescala para melhor deteção de objetos pequenos, de acordo com a estrutura definida na Figura 11. Ambos têm boa velocidade, mas apresentam problemas na deteção de pequenos objetos e desequilíbrio de categorias.

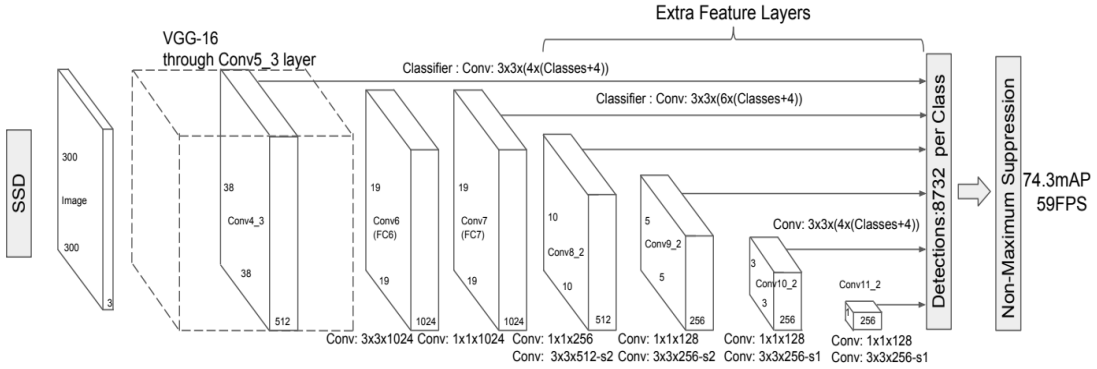


Figura 11 Estrutura SSD [92]

Como métodos de deteção de duas fases, destaca-se o R-CNN [103] que foi um dos primeiros detetores de objetos baseados em aprendizagem profunda e utiliza um algoritmo eficiente de Pesquisa Seletiva para obter propostas de região e o SPPNet [104]. Na rede R-CNN, a imagem inserida na rede CNN deve ser de tamanho fixo, o que significa que as propostas de região extraídas pelo algoritmo de Pesquisa Seletiva não são uniformes em tamanho e necessitam de passar por uma operação de distorção para uniformizar o tamanho, antes de serem inseridas na rede CNN. O problema com as diversas dimensões das imagens é que podem parecer artificiais e reduzir a precisão do reconhecimento [91].

O SPPNet acrescenta uma camada SPP após a camada convolucional para converter o mapa de características num vetor de características de comprimento fixo, que é depois inserido nas camadas totalmente conectadas. O SPPNet resolve não só o problema do tamanho das propostas de região como também o tempo de cálculo.

2.2.3.2 Rastreio Visual

Rastreio visual é um tópico que possui múltiplos desafios no campo da CV. No mundo real, o rastreio visual é influenciado por fatores externos, incluindo variações de pose, variações de iluminação, oclusões totais ou parciais e ruído no vídeo. Os investigadores da área dedicam especial atenção a métodos multi-sinais [75].

O rastreio visual visa a análise de sequência de vídeo para a localização de objetos em subsequência de quadros. Nas últimas décadas, uma série de algoritmos de rastreio de objetos foram propostos com o objetivo de vencer as dificuldades durante o rastreio [105].

Os métodos de rastreio utilizam diferentes sinais (cor, movimento, orientação, energia espacial, forma, textura, infravermelho, posição, etc.) foram amplamente detalhados e discutidos [106]. Quando se utiliza métodos de rastreio com múltiplos sinais, visa-se a localização do objeto através do processamento de dados adquiridos com sensores (unimodais ou multimodais), como demonstrado na Figura 12.

A falha de um sinal pode ser compensada por sinais complementares, o fluxograma de um rastreio multissensorial é descrito na Figura 13. Neste contexto, os sensores considerados para rastreio são ou um único sensor, como visão, infravermelho ou diferentes combinações de sensores, como visão, laser, radar, microfone, infravermelho, Microsoft Kinect e RFID.

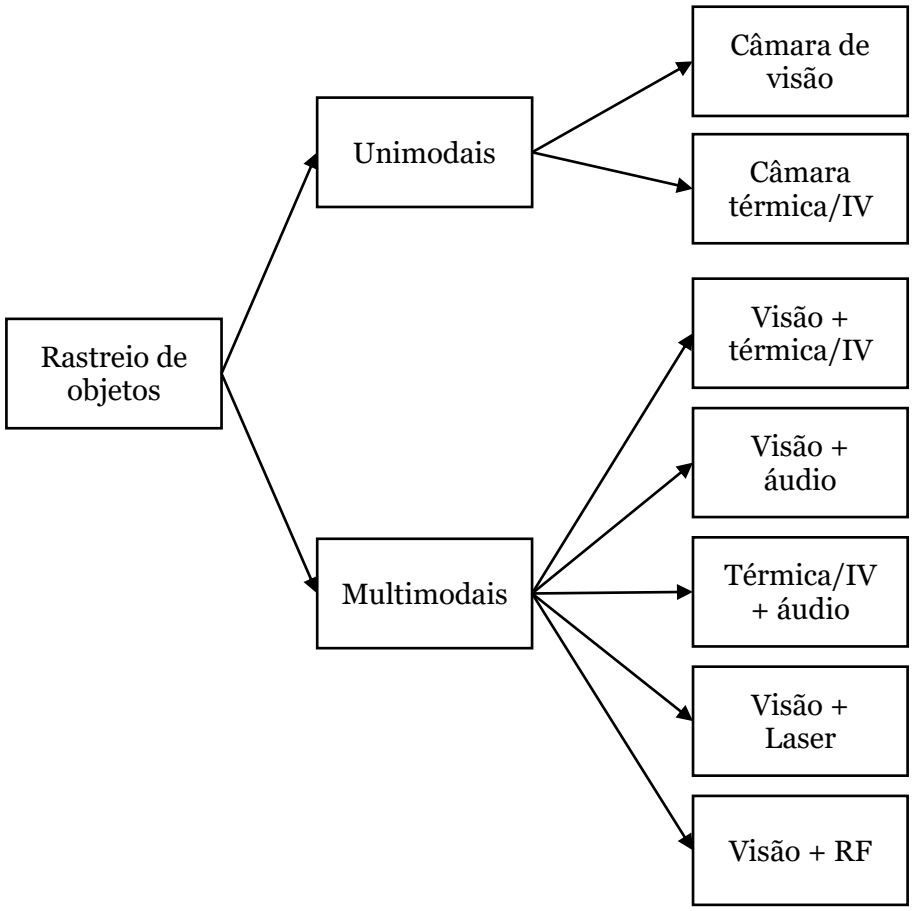


Figura 12 Classificação de técnicas de rastreio de objetos (Adaptado de [106]).

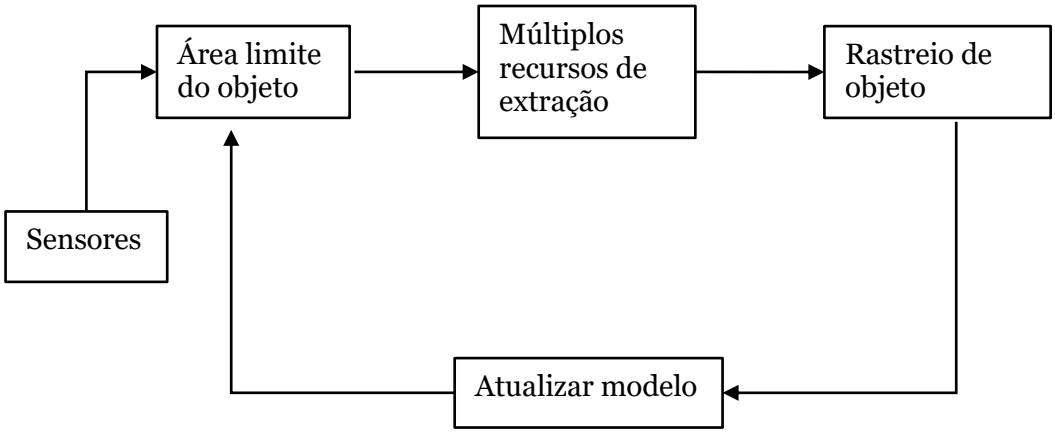


Figura 13 Diagrama de rastreio de objetos (Adaptado de [106]).

Muitos dos rastreadores existentes utilizam DL. O ADNet [107] destaca-se por ter sido desenvolvido com um rastreador que alcança tanto um cálculo leve quanto uma precisão de rastreio satisfatória. Este rastreador segue o alvo por meio de ações repetitivas controladas pela rede de decisão de ações, que é pré-treinada através de aprendizagem supervisionada e de reforço. Posteriormente, com a adaptação online no rastreio, o algoritmo de rastreio torna-se mais robusto contra deformações.

O GFS-DCF é um método que pode melhorar significativamente o desempenho de um rastreador DCF equipado com características de redes neuronais profundas. Esta abordagem envolve a criação de filtros correlativos para localizar um objeto em sequência de vídeo [75].

O D3S destaca-se entre os principais rastreadores da última geração, apesar de não ser treinado para rastreio de caixa delimitadora, tem um desempenho equiparável aos principais algoritmos de segmentação de objetos em vídeo, mas opera ordens de magnitude mais rápida. Vale salientar que o D3S não é treinado novamente para diferentes *benchmarks* – uma única versão pré-treinada mostra uma notável capacidade de generalização e versatilidade [108].

2.2.3.3 Segmento Semântico

Antes do surgimento dos modelos de redes neuronais, muitos métodos tradicionais foram desenvolvidos para resolver problemas de segmentação semântica. Algoritmos tradicionais representativos incluem métodos baseados em limiar, métodos baseados em agrupamento, transformadas de *wavelet* e floresta aleatória [109, 110]. Estes algoritmos tradicionais podem ser divididos em dois grupos principais, *Thresholding based e Clustering Based* [111].

No entanto, em outros domínios de classificação de imagens, a crescente predominância dos métodos baseados em CNN faz com que abordagens que recorrem a esta tecnologia se destaquem nesse campo. Os processos de segmentação semântica que empregam DL podem ser estruturados em três fases: extração de características, segmentação semântica e pós-processamento, tal como ilustrado na Figura 14 [111].

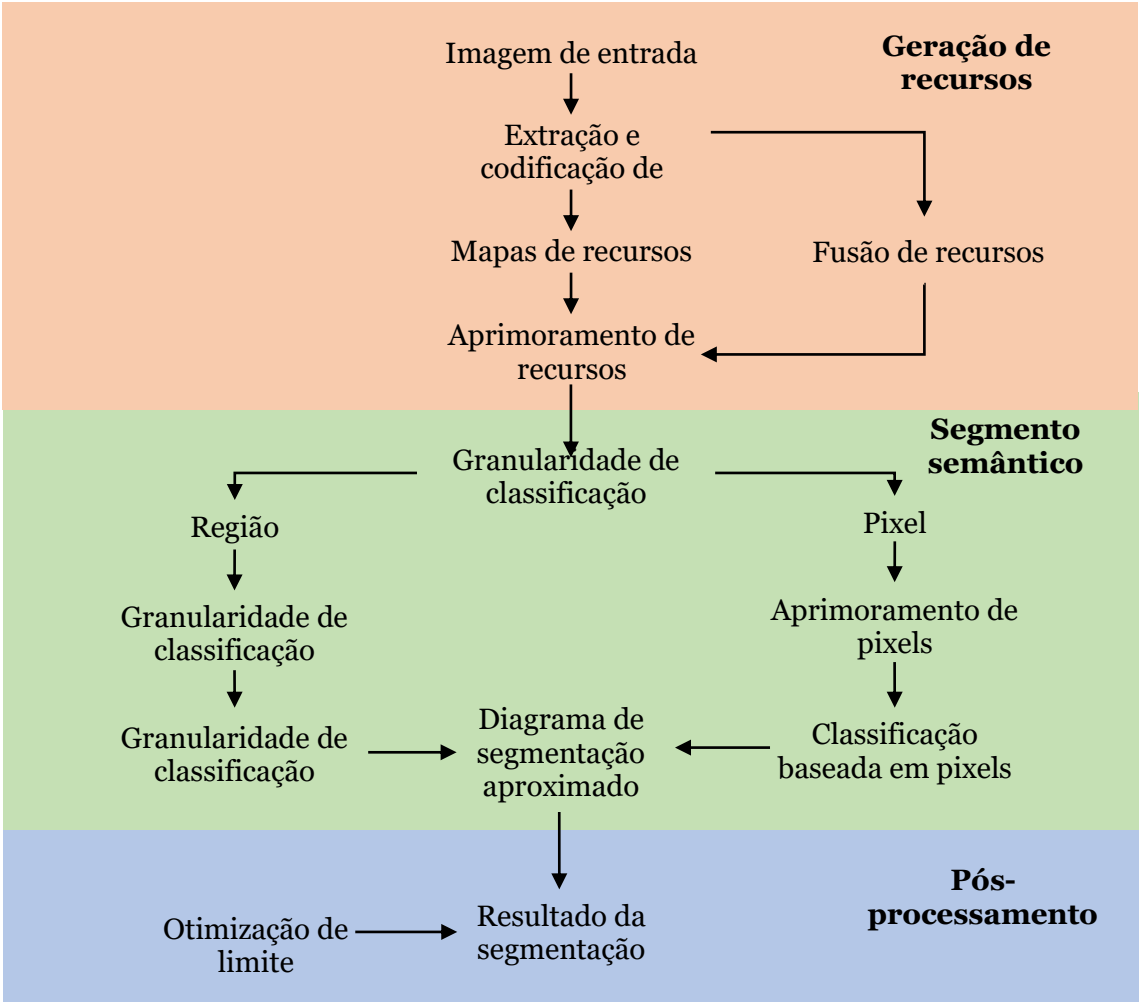


Figura 14 Processo de segmentação semântica baseado em DL [111].

A maioria das arquiteturas de modelos DL para segmentação semântica baseiam-se em estruturas *Enconder-decoder*. Entre os principais modelos que se utilizam dessa estruturam estão UNet++, DeepLabV3+. Também existem modelos que podem ser construídos com arquiteturas CNN usados especificamente para classificação de imagens, como as variantes ResNet e EfficientNet atuando como espinha dorsal [112].

A estrutura da arquitetura U-Net consiste em três componentes, uma fase de subamostragem que abstrai características nível por nível; uma fase de sobreamostragem que reconstrói as características; uma camada convolucional final para alcançar a classificação [113]. Devido à sua estrutura simplificada, a U-Net pode ser bem aplicada a imagens agrícolas com poucas amostras [111]. Além disso, a U-Net possui uma forte capacidade de extração de informações locais. Pode combinar bem as características dos detalhes globais e locais das imagens, e fundir informações de baixa resolução com regularidade e informações de alta resolução com limites desfocados e

gradientes complexos, o que pode alcançar uma segmentação precisa de algumas imagens agrícolas [114].

O DeepLabV1 [115] e o DeepLabV2 [116] adotam VGG e ResNet respetivamente como codificadores. O DeepLabV3 [117] ainda utiliza o ResNet como sua espinha dorsal. No entanto, o último bloco do ResNet é modificado com a adição de convolução [118]. A rede DeepLabV3+ [119] é excelente na extração de características mais densas. Ao mesmo tempo, o DeepLabV3+ expande largamente o campo de receção e obtém um fundo global multi-escala, mas ainda é limitado pela área local. Em segundo lugar, o enorme número de parâmetros traz uma tremenda carga computacional. Atualmente, a rede DeepLabV3+ é usada principalmente para alcançar segmentação de alta precisão de imagens de alta altitude [120,121,122].

2.2.3.4 Restauração de imagem

Ao lidar com técnicas de processamento de imagem, é necessário ter alguma ideia sobre a função de degradação, de modo que, durante o processo de restauração, o inverso da função de degradação possa ser aplicado para restaurar a imagem original. O processo de degradação é a incorporação da imagem original, a função de degradação h(x, y) juntamente com o ruído η(x, y) adicionado à função original. O processo de restauração é a medida da função de degradação e a aplicação da anti função dela. A Figura 15 ilustra o processo de degradação e restauração [123].

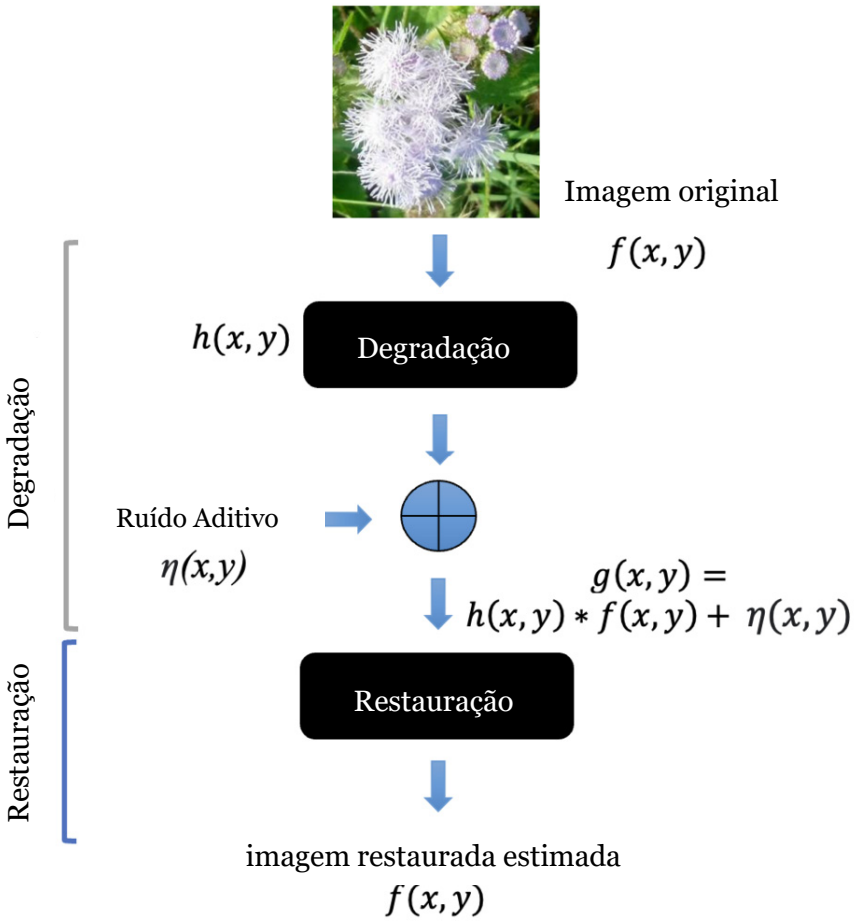


Figura 15 Modelo de degradação e restauração (Adaptado de [123]).

O reconhecimento de imagens ganhou novas perspetivas e aumentou significativamente sua popularidade com a introdução das CNNs em 2012 [124], Antes da introdução das CNNs, a maioria das abordagens de reconhecimento de imagens incluía processos de deteção e descrição de características feitas manualmente [125]. Múltiplas técnicas são introduzidas, como algoritmos de otimização para dados não convexos [126,127], técnicas que derivam as imagens a nível de pixel ou em regiões e, em seguida, combinadas no processo de reconhecimento [128].

Em contraste com métodos tradicionais de características feitas manualmente, como a transformação de característica invariante de escala (SIFT) [129], as CNNs detetam automaticamente as características da imagem por meio de múltiplas convoluções e operações não lineares. No entanto, a capacidade da CNN de aprender com as características relevantes depende inteiramente da capacidade de sua arquitetura [130].

O NOISE2VOID (N2V) [131] é uma abordagem de treino que requer aquisições "com ruido" únicas para treinar CNNs. Foi proposta uma rede de ponto cego, onde o campo recetivo de cada pixel é menos o próprio pixel, evitando assim aprender a identidade.

Assim, o N2V não pode remover bem o ruído, de modo que, se a suposição de independência não pudesse ser satisfeita, mostra uma nova maneira de treinar a rede para se adaptar aos campos que adquire com conjuntos de dados de resolução limitada ou baixa de forma realista. Outa abordagem é o CycleISP [132] que apresenta uma estrutura única que compreende duas ramificações de rede principais: RGB2RAW e RAW2RGB. Essas ramificações são projetadas para simular o processo ISP da câmara na conversão entre os formatos RGB e RAW, respetivamente. Uma rede auxiliar de correção de cores aprimora ainda mais a conversão RAW2RGB, garantindo que as imagens sRGB sintetizadas reproduzam fielmente as cores originais.

2.3 Utilização da visão computacional na indústria alimentar

Os sistemas de CV apresentam uma tecnologia não destrutiva, adequada para a classificação em linha e avaliação da qualidade de frutas e legumes. O sistema de CV mais comum para a inspeção de qualidade de produtos frescos é um sistema tradicional baseado em câmaras RGB que reproduzem a visão dos olhos humanos usando três filtros monocromáticos centrados nos comprimentos de onda vermelho (R - *Red*), verde (G - *Green*) e azul (B - *Blue*), com comprimentos de onda de 700, 546 e 435 nm, respetivamente [133]. Sistemas de CV baseado em imagens convencionais imitam a visão humana, adquirindo e analisando imagens da superfície visível do alimento para avaliar sua qualidade ou nível de maturidade, detetando características como cor, forma, tamanho ou textura. Detetando padrões de características mais distintivas relacionadas com a qualidade e classifica o produto usando modelos de classificação, como métodos clássicos de ML ou mais recentemente, DL.

Com os avanços em *hardware* e *software*, os sistemas de CV Hiper espectral [134] e multiespectral [135] foram desenvolvidos como tecnologias eficientes para a avaliação da qualidade de produtos agrícolas. Os dados de imagem espectral adquiridos pelos sistemas de CV Hiper espectral e multiespectral fornecem informações sobre características internas e externas que o sistema tradicional de CV tem dificuldade em avaliar. Estas técnicas fornecem uma avaliação de qualidade consistente e avaliação precisa da qualidade, mas a aquisição de imagens multiespectrais e híper espectrais produz uma grande quantidade de dados e requer dispositivos caros e complexos para aquisição e processamento. A sua aplicação geralmente requer competências específicas. O uso de sistemas tradicionais de CV pode proporcionar maior flexibilidade para corresponder ao custo, restrições de tempo e ambiente em diferentes pontos da cadeia de abastecimento até o utilizador final.

Entre os diversos tipos de sistemas tradicionais de CV, a escolha do método classificatório varia em função do produto analisado e do tipo de análise realizada. A Tabela 2 demonstra aplicações do sistema de visão computacional baseado em imagens convencionais.

Tabela 2 Aplicações de sistemas de CV em produtos hortofrutícolas

Produto hortofrutícola	Detalhes da Classificação	Métodos e Resultados de Avaliação	Referência
Amora	4 classes (amoras boas, enrugadas, deterioradas e danificadas mecanicamente) 2 orientações de frutas (extremidade do caule e extremidade do cálice)	Análise discriminante linear (LDA - *Linear Discriminant Analysis*) Máquina de vetores de suporte (SVM - *Support Vector Machine*) Identificação da orientação da fruta em 96,8% dos casos	Leiva-Valenzuela e Aguilera [136]
Beringela	4 categorias de saudáveis a não saudáveis (saudável, parcialmente defeituosa, moderadamente defeituosa e não saudável)	Vizinho mais próximo (KNN - *K-Nearest Neighbor*) (88% de precisão)	Akter e Rahman [137]
Espinafre fresco	Deteção de 4 graus, de bom a mau	Vizinho mais próximo (KNN) Máquina de vetores de suporte (SVM) Rede neuronal artificial de retro propagação (BPNN - *Back Propagation Neural Network*) (85,4% de precisão)	Huang et al. [138]
Laranja	Classificação automática em 3 grupos (grau A, B e C)	Rede neuronal artificial de retro propagação (94,4% de precisão geral, 100% para o grau A)	Chen et al. [139]
Maçã	Dois tipos de classificação: 2 categorias (saudável ou defeituosa). 3 classificações de qualidade (primeira classe, segunda classe e rejeitados)	Perceptora multicamada (MLP - *Multi Layer Perceptron*) Máquina de vetores de suporte (SVM) Vizinho mais próximo (KNN) 92,5% para 2 categorias 89,2% para 3 classificações de qualidade	Moallem et al. [140]
Manga	Identificação automática de defeitos: Deteção de maturidade	Pré-processamento por três algoritmos: Fruta podre acima do limite. Alta qualidade da fruta abaixo do limite	Sahu e Potdar [141]
Tomate	Classificação em defeituoso ou não defeituoso e maduro ou não maduro	Rede neuronal artificial 100% em tarefa defeituoso/não, defeituoso 96,5% na tarefa maduro/não, maduro	Arakeri e Lakshmana [142]

2.4 Casos de estudo

Neste subcapítulo, são explorados os casos de estudo que serviram de inspiração para esta Dissertação. Através de uma análise detalhada de exemplos práticos, desafios enfrentados, tecnologias utilizadas e metodologias aplicadas de casos semelhantes ao trabalho realizado.

2.4.1 Deteção automática de defeito múltiplo em batatas

O estudo desenvolvido por Yu *et al.* [61] propõe um método rápido e preciso para detetar automaticamente múltiplos tipos de defeitos em batatas, utilizando imagem multiespectral (MS) em conjunto com um modelo YOLOv3-tiny aprimorado. O sistema de imagem multiespectral (MSI) utilizado cobre 25 bandas de onda com uma resolução espacial de 409 × 216 pixels, adquirindo imagens de amostras de batatas, incluindo batatas sem defeitos e com defeitos (germinação, sarna comum, *bug-eye*, podridão seca e contusão), como apresentado na Figura 4.

Como materiais, para *hardware* foram utilizados câmara híper espectral tipo mosaico (modelo cmv2k-SNM-25-600-1000, Isuzu Optics, Taiwan, China), posicionada dentro de uma caixa preta e controlada por um computador com um CPU Intel(R) Core (TM) i7-6850K (3.60 GHz), uma GPU GeForce GTX 1080Ti de 11 GB (3584 núcleos CUDA), e 32 GB de memória. Duas lâmpadas de halogéno de fibra de 150 W serviram como sistema de iluminação, e uma esteira transportadora preta controlada por um sistema PLC foi usada para posicionar as batatas para as imagens.

As ferramentas de *software* utilizadas incluem CUDA 9.1, DUCNN 7.5, Python 3.6, e Linux Ubuntu 18.04. O YOLO-v5x é construído no *framework* Pytorch, e os outros três modelos são construídos nos *frameworks* Tensorflow e Keras.

Para a recolha de dados, foram separadas 428 batatas, 77 sem defeitos e 351 com defeito. Entre as amostras com defeitos, 118 casos de germinação, 219 de sarna comum, 59 de *bug-eye*, 83 de podridão seca e 80 de contusões. Todas as batatas foram limpas com água e secas por evaporação natural antes da aquisição de imagens. Durante a aquisição de uma imagem, a distância vertical entre o tapete transportador e a câmara era de 230 mm, colocando-se uma batata na interceção com a perpendicular à câmara. Para garantir a validade dos dados recolhidos, os defeitos da amostra defeituosa eram posicionados de frente para a câmara. O sistema MSI capturou simultaneamente informações espaciais e espectrais de uma superfície da amostra com curta exposição. Assim, todas as imagens MS foram obtidas com um tempo de exposição de 13 ms em condição estática da esteira.

Cada imagem MS era composta por 25 imagens de banda de onda (cobrindo de 676 a 952 nm) com um tamanho espacial de 409 × 216 pixéis e uma resolução espacial de 0,35 mm, ou seja, a dimensão de cada imagem MS era de 409 × 216 × 25.

O MDDNet, com uma estrutura semelhante ao YOLOv3-tiny, segue um processo de deteção para identificar defeitos em MS de batata, ilustrado na Figura 16.

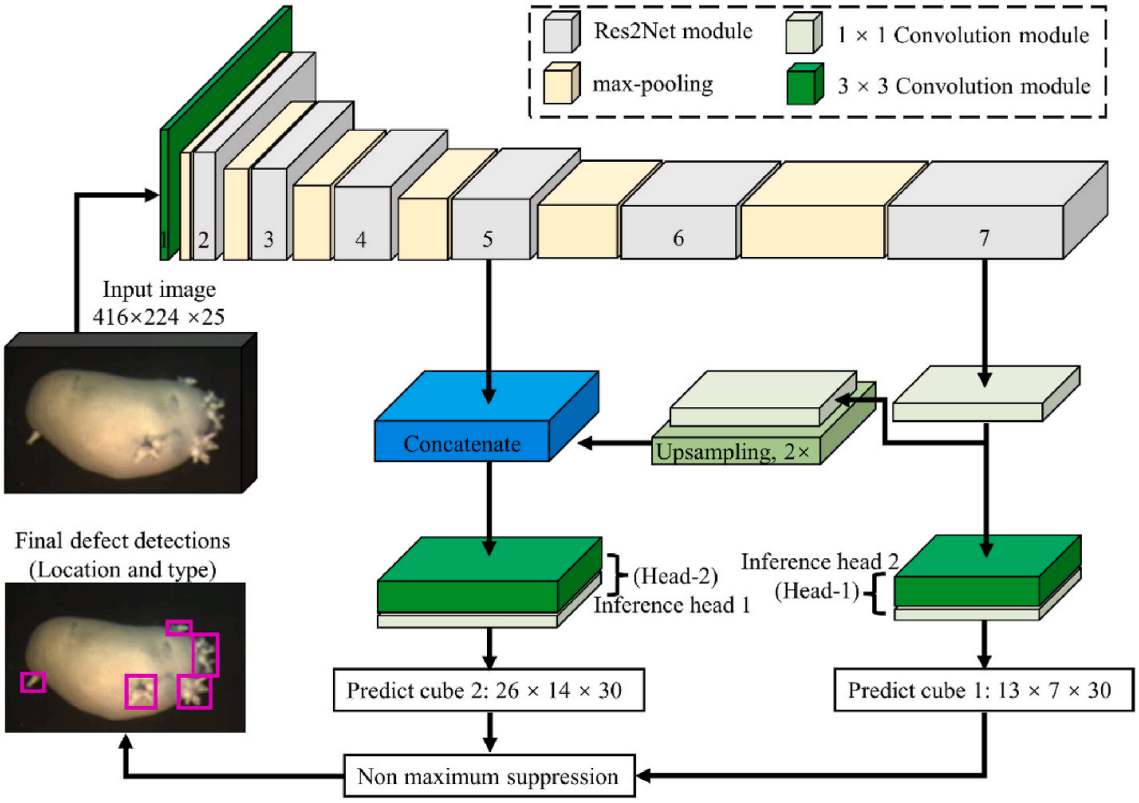


Figura 16 Detalhes da estrutura do MDDNet [61]

Durante o treino do modelo MDDNet por 400 iterações, foi utilizada a Inicialização de He et al. [143] para definir os pesos iniciais, que são otimizados pelo método de descida de gradiente estocástico (SGD) com momento de 0,9, tamanho de mini-lote de 20, taxa de aprendizagem de 0,001, limiar de ignorância de 0,5 e decaimento de peso de 0,0005, conforme a função de perda [144]. Cada módulo Res2Net divide o mapa de características de entrada em 4 grupos. Três métodos baseados em DL, nomeadamente YOLO-v5x, YOLOv3-tiny, e DY3TNet, são comparados com o MDDNet na deteção eficaz de defeitos em batatas, como demostrado na Figura 17. Cada rede é treinada e testada independentemente em um conjunto de imagens MS de batatas 10 vezes, e a média desses 10 testes é utilizada para avaliar justamente todos os métodos.As imagens MS foram manualmente etiquetadas com a ferramenta LabelImg, 70% das imagens, escolhidas de forma aleatórias, foram usadas para construir o conjunto de treino,

enquanto os 30% restantes formam o conjunto de teste.Para combater a escassez de dados, seis métodos comuns e eficazes de aumento de dados não supervisionados são aplicados para expandir o conjunto de treino. Estes incluem a inversão horizontal de todas as sequências de imagens, rotação de cada imagem bruta e a sua inversão em vários ângulos, escala de cada imagem bruta para 0,7–1,3 vezes o tamanho original e o ajuste da imagem escalada, translação de cada imagem bruta horizontalmente e verticalmente, adição de ruído gaussiano aleatório a todos os canais de cada pixel na imagem bruta, e inversão dos canais de cada imagem bruta. Ao todo, o conjunto de treino bruto foi aumentado dezassete vezes, resultando num total de 5400 imagens MS (300 imagens brutas e 5100 imagens aumentadas).A investigação avaliou cada modelo de deteção com medidas de precisão média (mAP), consumo de tempo e tamanho do modelo. Os resultados experimentais mostraram que o modelo MDDNet, com pesos ocupando 90,76 MB de armazenamento, obteve o mAP mais alto de 90,23% e manteve uma velocidade de deteção de 75 ms por imagem. Pode-se concluir que o sistema MSI, combinado com o modelo MDDNet, consegue detetar com sucesso defeitos de múltiplos tipos em batatas.

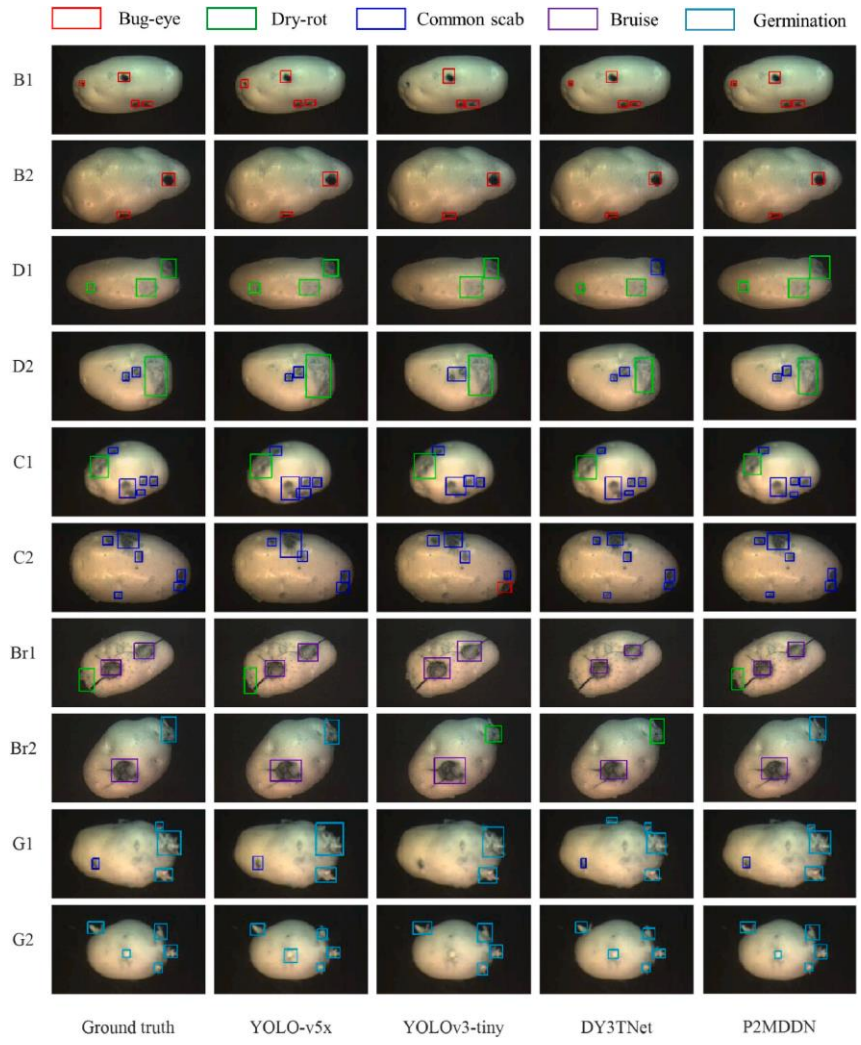


Figura 17 Exemplos de deteção de imagem dos quatro modelos de aprendizagem profunda [61].

A principal mais-valia deste estudo para a presente dissertação reside na sua recolha, gestão e processamento de dados. Extrair-se-á deste trabalho metodologias de recolha e organização de dados, como a quantidade de imagens de batatas que foram adquiridas, a quantidade de batatas saudáveis e não saudáveis, a ampliação do número de dados através de métodos tradicionais, a etiquetagem através da ferramenta Labelling de maneira manual e a separação 70-30 para conjunto de treino e teste. Além disso, as técnicas empregues no treino do modelo computacional serão de particular interesse, para demostrar a eficácia da inicialização de He et al [143] para trabalhos de deteção de falhas em hortofrutícolas.

2.4.2 Sistema de deteção de objetos para pessoas com deficiência visual

O estudo desenvolvido por Bin *et al.* [145] implementa uma abordagem para ajudar pessoas com deficiências visuais a navegar de forma independente no ambiente, através da execução de dois modos: deteção de objetos e descrição do ambiente envolvente.

O artigo utiliza de um conjunto de dados MS COCO (*Microsoft Common Objects in Context*) que possui 165.000 imagens, rotuladas em 90 classes.

Como *hardware* foi utilizado o Raspberry Pi 4 Modelo B+ como dispositivo principal, a câmara Raspberry Pi v2 para captura de vídeo, e uma *power bank* de 10000 mAh como fonte de alimentação. A saída de voz é transmitida através de fones de ouvido, o sistema final pode ser observado na Figura 18.

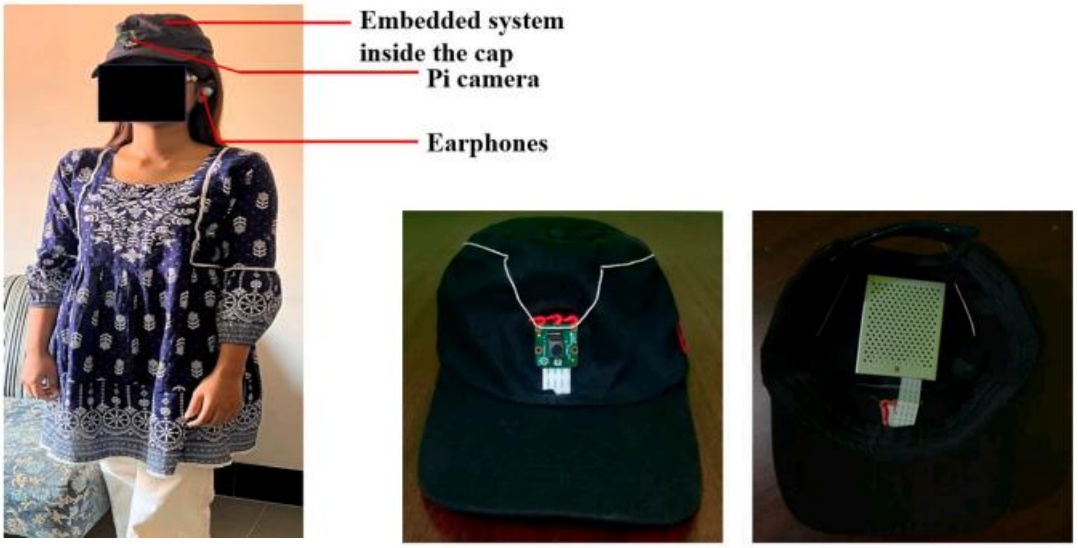


Figura 18 Configuração de teste do utilizador do dispositivo proposto. [145]

Tanto para deteção de objetos como para o modo ambiente foi utilizado um modelo baseado em redes neuronais profundas SSD, com a variação MobileNetV2 [146].

Para deteção de objetos, foram realizados ensaios em tempo real (interiores e exteriores) com distância aproximada de 0,05 m a 3 m, obtendo-se resultados de precisão de 88,89% com uma frequência média de 2,15 FPS.

A principal mais-valia deste estudo para a presente dissertação é o uso de um *hardware* de baixo custo, semelhante ao que se pretende utilizar: um RaspBerry Pi como microcontrolador e uma Pi Câmara para a captura de vídeo, para além de usar o mesmo modelo de deteção de objetos, o MobileNetV2 e sua adaptação para o TensorFlow Lite, permitindo assim correr os modelos treinados com as capacidades de processamento mais limitadas, no *hardware* disponível.

2.4.3 Manipulação controlada por visão computacional em robô industrial

O trabalho de Andhare et al.[147] apresenta uma investigação no campo da aprendizagem de máquina e robótica, focando-se em vários aspetos como o processamento visual, transformação de coordenadas, cálculo de posição, algoritmo de orientação, e solução de diretrizes para robôs.

O *hardware* utilizado foi um braço robótico com 6 DOF, uma câmara e um computador para o controlo.

A identificação de objetos é feita através da programação em Python com OpenCV, empregando deteção de bordas para classificar objetos com base na área do contorno. A orientação e a posição dos objetos são determinadas para a manipulação pelo robô, que os classifica e coloca em caixas apropriadas dependendo do seu tamanho. A comunicação entre o sistema de visão e o controlador do robô é realizada via Ethernet, utilizando comunicação por pacotes para transmitir coordenadas dos objetos à distância, facilitando a integração entre visão e ação mecânica. Além disso, é desenvolvido um algoritmo de orientação com base na área mínima em torno do objeto para direcionar a ação do manipulador. A lógica do algoritmo está ilustrada na Figura 19.

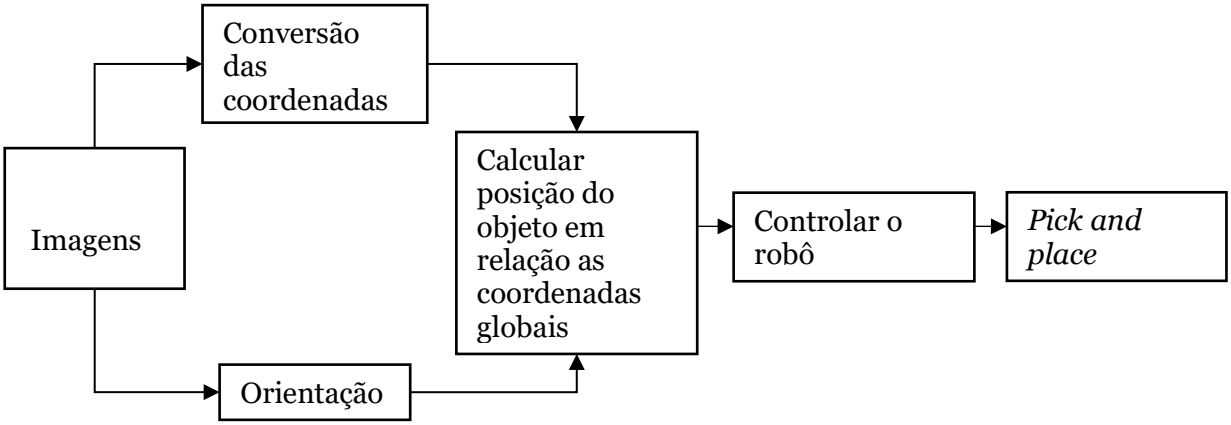


Figura 19 Lógica do algoritmo do estudo (Adaptado de [147]).

A principal mais-valia deste estudo para esta dissertação é a sua lógica de controlo dos robôs com CV, separando uma área de interesse principal apesar de que neste estudo é utilizado *hardware* diferente do que é utilizado nesta dissertação. No caso neste estudo é utilizado um robô com 6 DOF e um computador para o controlo. Porém, adaptar a ideia do controlo através do algoritmo representado na Figura 19 e sua implementação de coordenadas globais é completamente possível.

2.5 Nota Conclusiva

Este capítulo descreve a influência das revoluções industriais no setor alimentar, apresentando tecnologias que transformaram esta indústria, dando enfase a CV e AI. A evolução da CV foi analisada abordando deste seus fundamentos e métodos clássicos a utilização de tecnologias DL, também são exploradas neste capítulo aplicações usuais desta tecnologia, como classificação de imagem, deteção de objetos, rastreio visual, segmentação semântica e restauração de imagens.

Entender os conceitos de CV e as suas implicações é essencial para entender como esta tecnologia se tornou uma ferramenta útil na indústria alimentar, sendo um método não destrutivo, barato e de fácil implementação.

Este capítulo ainda apresenta diversas utilizações desta tecnologia na indústria, através da análise de estudos científicos que implementaram sua utilização em diversas frutas e vegetais. São destacados casos de estudo que se tornaram pilares fundamentais para a base científica desta dissertação.

Capítulo 3

Materiais e Métodos

Neste capítulo, são discutidos e apresentados os métodos aplicados e os materiais utilizados nesta dissertação. Resumidamente, o projeto consiste na utilização de um Raspberry Pi 5 acoplado a um suporte, permitindo a captura de imagens dos frutos por meio de uma PiCamera módulo 3. As imagens são enviadas via cabo Ethernet para um braço robótico UR3e da Universal Robots, utilizando um código Python que integra funções modulares fornecidas pela empresa e funções desenvolvidas especificamente para esta dissertação.

O braço robótico está equipado com o atuador 2F-85 da Robotiq com dedos macios produzidos para o projeto, permitindo a manipulação de fruta delicada sem causar danos, de acordo com a metodologia não destrutiva adotada. Para o treino do modelo de deteção de frutos, foi utilizado o TensorFlow Object Detection API, seguido pela conversão para TensorFlow Lite, que será o modelo final implementado.

Os dados de posição obtidos pelo modelo são transmitidos ao braço robótico, que então manipula as frutas para o local definido, através de uma operação eficiente e precisa.

3.1 Braço robótico

Um braço robótico é desenhado e projetado para que se assemelhe a um braço humano, composto por ligações interligadas em série ou paralelo, pode ser controlado programando-o para executar uma tarefa específica [148]. As juntas do manipulador conectam os elos que levam ao deslocamento, que pode ser linear (juntas prismática) ou rotacional (juntas rotacionais) [142], a diferença entre esses movimentos está ilustrada na Figura 20.

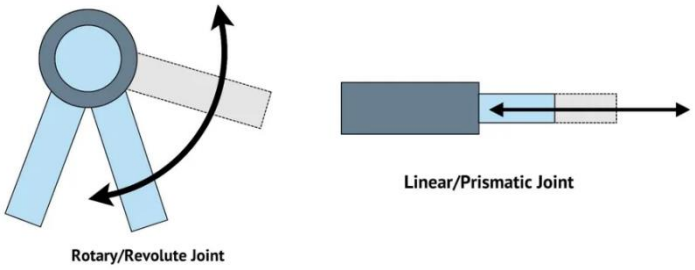


Figura 20 Movimento linear vs. Rotacional [143].

A geometria de um manipulador robótico é convenientemente definida anexando referenciais a cada elo. Enquanto esses referenciais podem ser localizados arbitrariamente, é vantajoso tanto para consistência quanto para eficiência computacional aderir a uma convenção para localizar a posição dos elos [144]. A convenção definida por Denavit e Hartenberg (D-H) é uma descrição consistente e concisa da cinemática da relação entre os elos de uma cadeia cinemática [145].

3.1.1 Convenção Denavit-Hartenberg

O processo de identificação de parâmetros necessita de uma descrição precisa dos referenciais ligados às componentes da cadeia cinemática. Deve-se proceder através das seguintes etapas:

1. Identificar elos e juntas: os elos são numerados de 0 (base) a n (ferramenta). As juntas são numeras de 1 a n.

2. Definir os quadros de referência para os elos internos: Localizar o eixo Z_i ao longo do eixo da junta i+1. A origem do quadro 0_i está posicionada ao longo do eixo da junta i+1. Se os eixos Z são paralelos, 0_i é escolhido arbitrariamente. Caso contrário, ele está localizado na interseção entre Z_i e a normal comum a Z_{i-1}e Z_i. Y_i é o escolhido para compor um quadro à direita.

3. Definir os referenciais para os elos das extremidades: Z_0 está localizado ao longo do eixo da junta 1. X_0 e Y_0 são arbitrários: eixos X_né normal ao eixo do conjunto n, enquanto Y_ne Z_n são definidas arbitrariamente.

4. Identificar os parâmetros D-H para cada elo. a_i é a distância entre Z_{i-1} e Z_i, d_i é a distância entre x_{i-1} e x_i, α_i é o ângulo entre Z_{i-1} e Z_i medido ao longo de x_i e θ_i é o ângulo entre x_{i-1} e x_i, medido ao longo de Z_i.

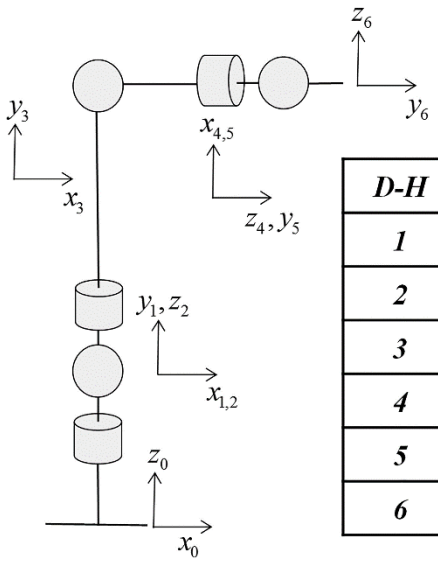
5. Determinar as matrizes de transformação homogéneas para cada junta.

6. Determinar a matriz de transformação homogénea geral por pré-multiplicação das matrizes de transformação conjunta individuais

A matriz transformação está descrita na equação 1.

$$A_i^{i-1}(q_i) = A_{i'}^{i-1}A_i^{i'} = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Um exemplo da aplicação dos parâmetros D-H, pode ser observado no manipulador DLR, ilustrado na Figura 21.



D-H	q	d	a	α
1	q_1	d_1	0	$\pi/2$
2	q_2	0	0	$-\pi/2$
3	q_3	d_3	0	$\pi/2$
4	$q_4+\pi/2$	0	0	$\pi/2$
5	q_5	d_5	0	$\pi/2$
6	$q_6+\pi/2$	0	a_6	$\pi/2$

Figura 21 Aplicação dos parâmetros DH em um braço com 6 DOF [146]

Com a aplicação desta convenção, é possível aplicar métodos cinemáticos de controlo para definir o ponto da ferramenta no momento desejado, como utilizando a cinemática direta, este modo de controlo pode ser descrito pela equação 2.

$$T_n^0(q) = A_1^0(q1)A_2^1(q2) \ldots . A_n^{n-1} \quad (2)$$

3.1.2 Cinemática Inversa

A cinemática inversa consiste na determinação das variáveis articulares correspondentes a uma dada posição e orientação do posição e orientação do atuador final. A solução deste problema é de importância fundamental para transformar as especificações de movimento, atribuídas ao atuador final no espaço operacional, nos movimentos correspondentes no espaço articular que permitem a execução do movimento desejado.

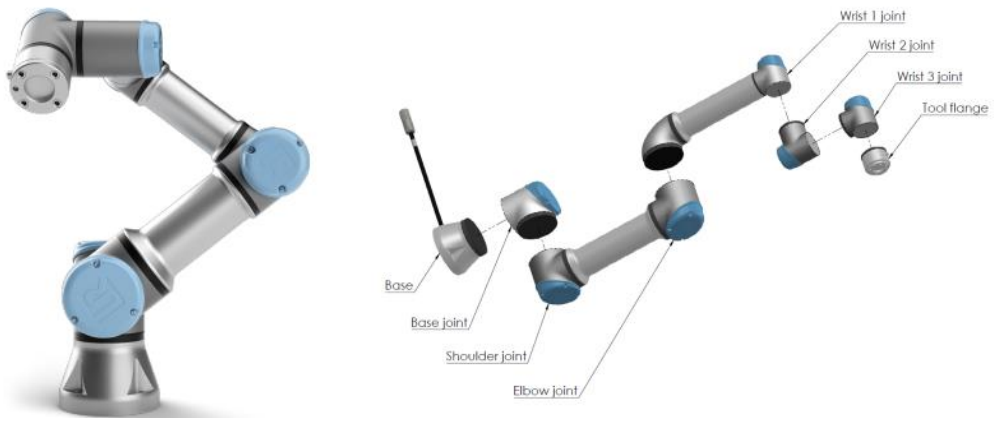
No que diz respeito à equação da cinemática direta a posição do vetor final e a matriz de rotação são calculadas de uma forma única, uma vez conhecidas as variáveis articulares. No que diz respeito à equação da cinemática direta, a posição do vetor final e a matriz de rotação são calculadas de uma forma única, uma vez conhecidas as variáveis articulares. Por outro lado, o problema da cinemática inversa é mais complexo, devido:

1. As equações a resolver são, em geral, não lineares, pelo que nem sempre é possível encontrar uma solução em forma fechada.
2. Podem existir várias soluções.
3. Podem existir soluções infinitas, por exemplo, no caso de um manipulador cinemático redundante.
4. Pode não haver soluções admissíveis, tendo em conta a estrutura cinemática do manipulador

A existência de soluções só é garantida se a posição e a orientação do vetor final pertencerem ao envelope de trabalho do manipulador.
Por outro lado, o problema das soluções múltiplas depende não só do número DoF mas também do número de parâmetros não-nulos da convenção DH. Em geral, quanto maior o número de parâmetros não-nulos, maior o número de soluções admissíveis. Para um manipulador de seis DoFs sem limites mecânicos de junta, existem em geral até 16 soluções admissíveis. O cálculo de soluções de forma fechada requer cálculo algébrico para encontrar as equações significativas que contêm as incógnitas ou análise geométrica para encontrar os pontos significativos na estrutura em relação aos quais é conveniente expressar a posição e/ou orientação como uma função de um número reduzido de incógnitas [144].

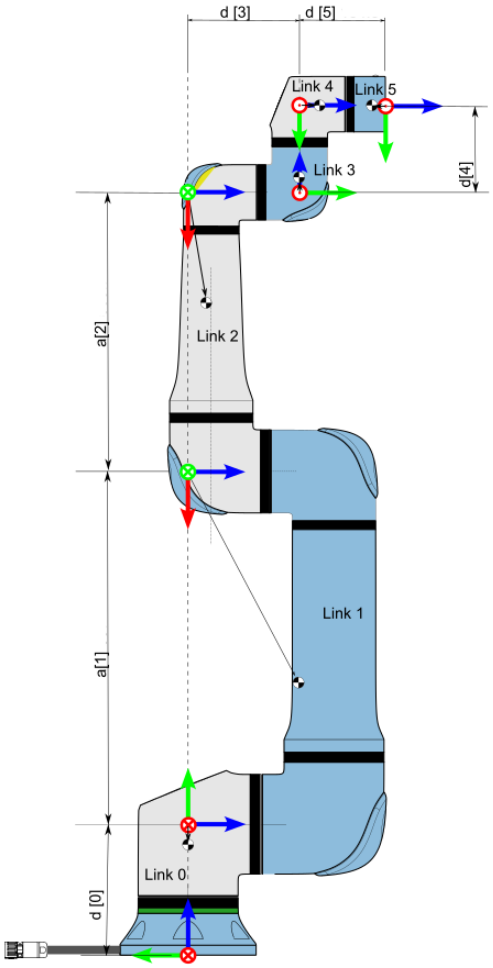
3.1.3 UR3e

O UR3e da Universal Robots, representado na Figura 22 de 3 kg, proposto para a automação de tarefas que não exijam alcances nem cargas elevadas [153]. No geral, as principais aplicações do UR3e são tarefas de manipulação (*pick-and-place*), soldadura, corte, desbaste, polimento, pintura, dosagem, fabrico aditivo e fresagem. A aplicação da convenção DH no UR3e está ilustrada na Figura 23.



(a) (b)

Figura 22 UR3e montado (a), e separado em juntas e elos (b) [154].



Cinemática	theta [rad]	a [m]	d [m]	alpha [rad]	Dinâmica	Massa [kg]	Centro de Massa [m]	Matriz de Inércia
Articulação 1	0	0	0,15185	π/2	Elo 1	1,98	[0, -0.02, 0]	0
Articulação 2	0	-0,24355	0	0	Elo 2	3,4445	[0.13, 0, 0.1157]	0
Articulação 3	0	-0,2132	0	0	Elo 3	1,437	[0.05, 0, 0.0238]	0
Articulação 4	0	0	0,13105	π/2	Elo 4	0,871	[0, 0, 0.01]	0
Articulação 5	0	0,08535	0	-π/2	Elo 5	0,805	[0, 0, 0]	0
Articulação 6	0	0	0,0921	0	Elo 6	0,261	[0, 0, -0.02]	[0, 0, 0, 0, 0, 0, 0.0001]

Figura 23 Aplicação da convenção DH no UR3e [155].

3.1.4 Atuador 2F-85

A garra adaptativa Robotiq 2F-85, representada na Figura 24, é um atuador desenvolvido para robôs colaborativos, projetado para oferecer uma gama completa de aplicações com um rápido tempo de produção.

Figura 24 Garra Robotiq 2F-85

As especificações técnicas da garra são apresentadas na Tabela 3.

Tabela 3 Especificações técnicas da garra 2F-85

Parâmetro	Especificação
Curso	85 mm
Força de preensão	20 a 235 N
Carga útil do punho de encaixe	5 kg
Carga útil do punho de fricção	5 kg
Peso da pinça	0.9 kg
Velocidade de fecho	20 to 150 mm/s
Classificação de proteção de entrada (IP)	IP40

3.1.3.1 Garra desenvolvida para o atuador

Implementou-se um modelo de garras flexíveis que utilizam o efeito barbatana (*Fin Ray Effect* – FRE), representado na Figura 25. Estes modelos de são inspirados biologicamente nas barbatanas de peixes e são compostos por dois feixes longitudinais macios conectados de forma simétrica numa configuração em V, com diversos travessões de comprimentos crescentes espaçados entre a ponta e a base [156]. Foi escolhido o desenvolvimento desses manipuladores macios pois apresentam uma abordagem mais simples e economicamente viável para agarrar objetos de diversas formas [157]. Além disso, o uso de garras flexíveis permite uma melhor adaptação ao produto, reduzindo a pressão de manipulação sobre a superfície do objeto.

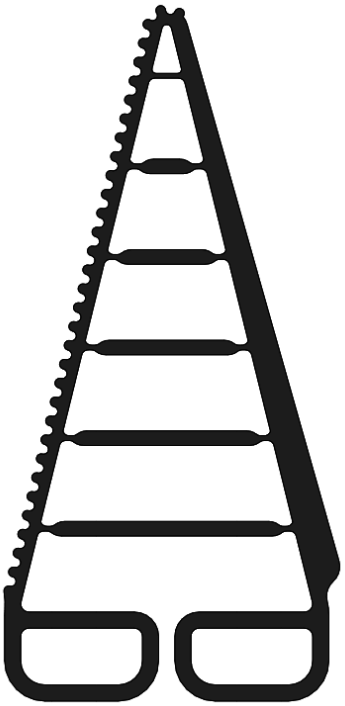
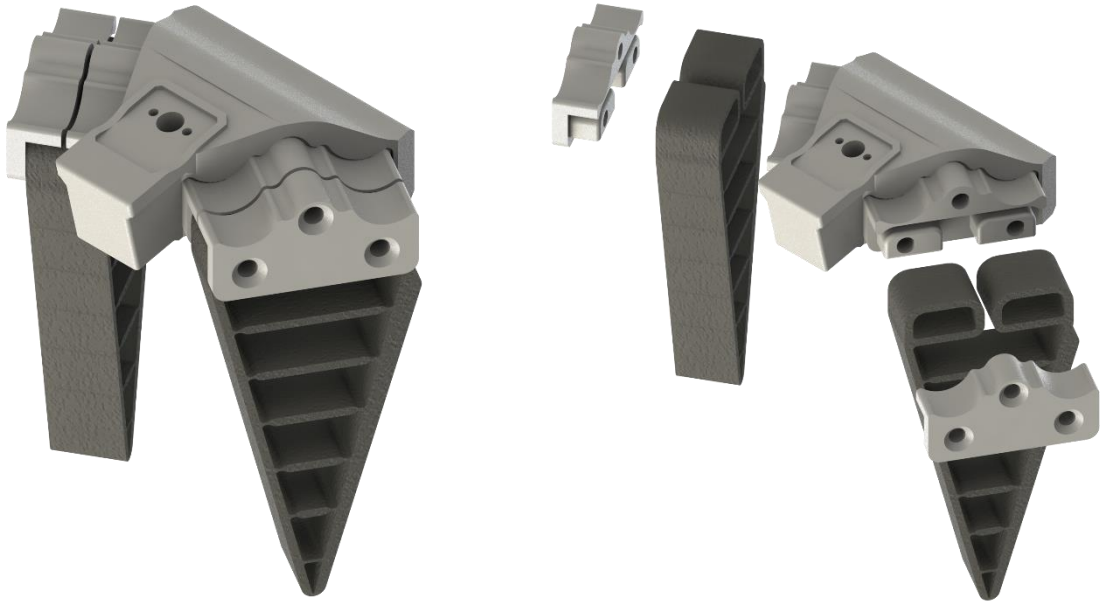


Figura 25 Desenho CAD de um dedo da garra

Para isso, as garras foram projetadas em CAD utilizando o *software* SolidWorks para produção por fabrico aditivo. Em seguida, o CAM foi realizado com o *software* Cura, gerando o código de máquina (G-CODE) necessário para a impressão numa impressora 3D, especificamente a Artillery Sidewinder X1 [151].

O material escolhido foi o TPU Filaflex 60A da empresa Recreus, devido à sua alta flexibilidade e elasticidade, sendo o filamento mais flexível disponível no mercado atual. Este material tem um custo reduzido e baixa rigidez, facilitando a manipulação de frutos sensíveis, como o tomate, sem causar danos.

A garra é acoplada num adaptador, formado por duas peças, desenvolvidos para integrar o atuador final da Robotiq, que permite o encaixe com um parafuso M5. O adaptador é formado por duas peças, impressas em PETG e apresentadas na Figura 26 (a) e (b).



(a) (b)

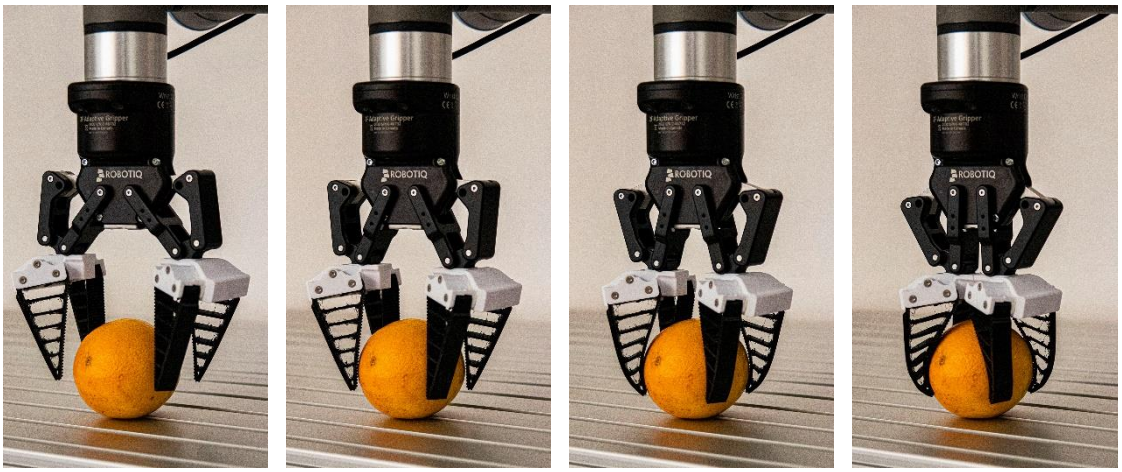
Figura 26 Imagem renderizada do dedo duplo FRE que substitui os dedos originais da garra da Robotiq, e (b) vista explodida do dedo duplo com as componentes que o constituem.

A Figura 27 apresenta a montagem destes dedos na garra da Robotiq, finalizando a montagem da garra. A escolha de uma garra com quatro dedos visa melhorar a precisão, independentemente do tamanho do fruto a ser manipulado. Este formato também oferece maior margem de erro no controlo do braço. Assim, se o fruto se mover durante o processo de manipulação, o design da garra compensará esse movimento.

(a) (b)

Figura 27 Processo de montagem dos dedos FRE na garra com dois parafusos M5 (a) e garra montada (b)

A Figura 28 apresenta o processo de agarrar uma laranja com os dedos FRE



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

Figura 28 Sequência de imagens que ilustra o FRE desde o ponto de abertura máxima da garra (a), passando pela transição (b) e (c) até ao fecho (d), para agarrar uma laranja

3.2 Raspberry Pi

Com uma produção ultrapassando um milhão de unidades por mês em 2023 [158], o Raspberry Pi cada vez mais se comprova como uma referência do mercado de microcontroladores. O seu tamanho diminuto e a capacidade computacional capaz de executar aplicações de computadores tradicionais é muito relevante [159]. Originalmente desenvolvido para incentivar programação [160], tornou-se uma ferramenta de controlo programável para funções M2M (*Machine to Machine*).

Existem vários *hardwares* de apoio produzidos pela Raspberry Pi Foundation, desde ratos e teclados até kits de antenas, câmaras, displays com touch, coolers.

3.2.1 Raspberry Pi 5

Com um processador Arm Cortex-A76 quad-core de 64 bits a 2.4 GHz, o mais recente modelo da Raspberry Pi Foundation oferece um aumento de 2 a 3 vezes no desempenho da CPU em relação ao mais moderno modelo da família Raspberry Pi 4 e um aumento substancial no desempenho gráfico com uma GPU VideoCore VII de 800 MHz. Este modelo está equipado com o RP1, um controlador I/O constituído de silício construído internamente no Raspberry Pi. A Figura 29 ilustra em detalhes o Raspberry Pi 5.

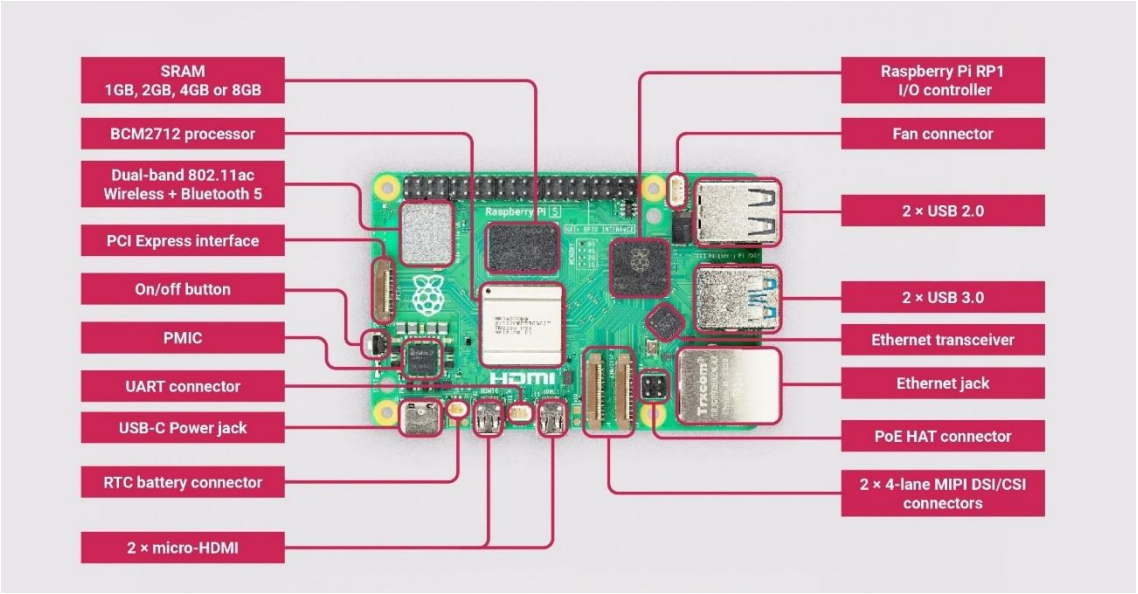


Figura 29 Raspberry pi 5 [161].

As dimensões do Raspberry Pi 5 são apresentadas no desenho técnico da Figura 30.

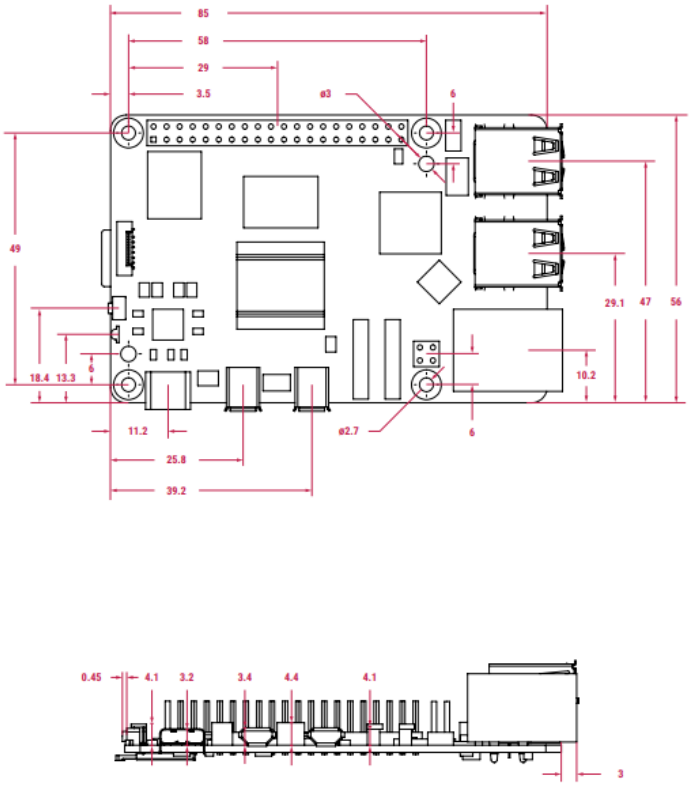


Figura 30 Raspberry Pi 5 desenho mecânico [162]

Estas dimensões serão utilizadas mais tarde para a construção do sistema de fixação do microcontrolador e da câmara.

3.2.2 PiCamera módulo 3

Com 12 megapixéis, o terceiro módulo oficial das câmaras Raspberry Pi supera em 4 megapixéis o modelo anterior. Esta câmara liga-se ao Raspberry por um cabo flexível a uma entrada CSI, tal como se pode ver na Figura 31.

Figura 31 PiCamera ligada a um Raspberry Pi 4 [163].

Estas câmaras podem ser programas por uma biblioteca em Python chamada Picamera2. Esta biblioteca é construída para códigos abertos libcamera, que fornece suporte para sistemas complexos de câmaras em Linux. O Picamera2 usa diretamente as ligações Python fornecidas pela libcamera. Existem 2 principais tipos de PiCamera, a sua versão padrão e a *wide*, sendo que cada tipo existe a versão que utiliza um filtro de corte infravermelho e a que não utiliza este filtro (NoIR). A especificação de cada PiCamera está descrita na Tabela 4.

Tabela 4 Caracterização das diferentes versões da PiCamera Module 3 [163].

Especificação técnica	**Camera Module 3**	**Camera Module 3 NoIR**	**Camera Module 3 Wide**	**Camera Module 3 Wide NoIR**
Gama de focagem [mm]	100–∞	100–∞	50–∞	50–∞
Distância focal [mm]	4.74	4.74	2.75	2.75
Campo de visão diagonal [º]	75	75	120	120
Campo de visão horizontal [º]	66	66	102	102
Campo de visão vertical [º]	41	41	67	67
Relação focal (F-stop)	F1.8	F1.8	F2.2	F2.2
Sensível aos infravermelhos	Não	Sim	Não	Sim

3.3 Ferramentas para visão computacional

Como discutido na secção 2.2, a CV, principalmente após a implementação dos métodos de DL, tornou-se parte ativa da automação e deteção em diversas atividades, industriais ou não. Deste modo, múltiplas ferramentas destacaram-se em implementar o uso dessa tecnologia não invasiva e extremamente útil ao mercado no geral. Nesta secção são destacadas as ferramentas escolhias para o estudo desta dissertação assim como a justificação para a sua escolha.

3.3.1 Linguagem de programação Python

A linguagem Python, que tem como logo a representação da Figura 32 foi criada em 1990 por Guido Van Rossum, inicialmente dirigida para utilizadores como físicos e engenheiros. Uma das principais características do Python é ser de código aberto, o que, aliado à sua sintaxe simples, torna-a uma opção sólida para diversas aplicações. Como resultado, em 2024, Python tornou-se na linguagem mais utilizada no mundo [164]. Dessa forma, esta linguagem possui uma grande comunidade *online* que desenvolve diversas bibliotecas focadas em diferentes áreas profissionais. Outra vantagem do Python é sua capacidade de integrar código de outras linguagens, facilitando a execução de múltiplas funções. Isso permite que a interpretação seja feita linha a linha, simplificando a gestão de memória e CPU e tornando a depuração mais acessível

Figura 32 Logo Python

3.3.2 OpenCv

OpenCV (*Open Source Computer Vision*) é uma biblioteca de programação, de código aberto com uma variedade de ferramentas de interpretação de imagens, sua logo é representada na Figura 33, indo desde operações simples como um filtro de ruído, até operações complexas, tais como a análise de movimentos, reconhecimento de padrões e reconstrução em 3D. A biblioteca OpenCV foi desenvolvida pela Intel e possui mais de 500 funções [165]. A biblioteca está dividida em cinco grupos de funções: Processamento de imagens; Análise estrutural; Análise de movimento e rastreio de objetos; Reconhecimento de padrões e Calibração de câmara e reconstrução 3D. O OpenCV

possui a maior biblioteca de visão computacional [166], tornando uma escolha confiável para este estudo.

Neste projeto, o OpenCV foi utilizado para inicialização e exibição do FPS, conversão da imagem de BGR (*Blue-Green-Red*) para RGB (*Red-Green- Blue*), redimensionar o quadro para o esperado pelo modelo computacional, normalização dos valores dos pixéis, desenhar as caixas delimitadora, exibir e calcular o ponto central de cada objeto detetado em tempo real.

Figura 33 Logo OpenCV

3.3.3 ArUco

Um marcador ArUco é um marcador quadrado sintético composto por uma borda preta larga e uma matriz binária interna que determina o seu identificador, um exemplo é demostrado pela Figura 34. A borda preta facilita a sua rápida deteção na imagem e a codificação binária permite sua identificação e a aplicação de técnicas de deteção e correção de erros. O tamanho do marcador determina o tamanho da matriz interna. Por exemplo, um marcador de tamanho 4x4 é composto por 16 bits.

Os marcadores ArUco são um tipo de marcador fiável amplamente utilizado na visão computacional. Este marcador têm o objetivo de fornecer pontos de referência visual confiáveis, permitindo a localização e a orientação precisa de objetos ou de câmaras num ambiente. São geralmente utilizados para calibração de câmaras e estimativa de tamanho do objeto.

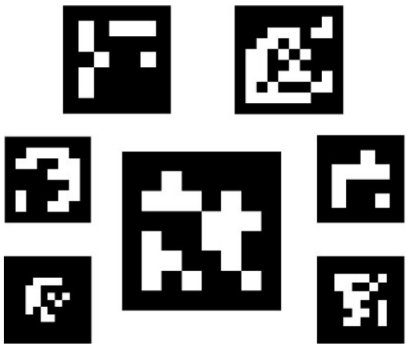


Figura 34 Marcadores ArUco

3.3.4 TensorFlow

TensorFlow (TF), que tem como logo a Figura 35, é uma biblioteca flexível e escalável para cálculos numéricos usando gráficos de fluxo de dados. Essa biblioteca e ferramentas relacionadas permitem o treino com eficiência de redes neuronais e outros modelos de aprendizagem máquina. Possui APIs em diversas linguagens, sendo a mais completa e estável o Python [167]. TF foi criado e é mantido pela Google. Durante o projeto, a Google desenvolveu o TF para poder correr em sistemas heterogéneos, incluindo dispositivos móveis. A primeira versão dessa evolução foi o TF Mobile, que foi substituído pelo TF Lite, que suporta a implementação para dispositivos móveis. Algumas das otimizações incluem aceleração de *hardware* através da camada de silício, estruturas como Android Neuronal e API de rede de neuronais artificiais otimizadas para dispositivos móveis, como MobileNets. O TF disponibiliza ferramentas para converter modelos de TF para TFLite [168].

Neste projeto, o TF foi utilizado para treinar o modelo, primeiramente em TF e logo depois convertido em TFLite, ideal para ser utilizado em uma plataforma como o Raspberry Pi, e como ferramenta de verificação da qualidade do treino do modelo.

Figura 35 Logo TensorFlow

3.3.4.1 TensorFlow Object Detection API

A TensorFlow Object Detection API é uma ferramenta desenvolvida pela Google, que permite construir, treinar e implementar sistemas de deteção de objetos de imagens. A API utiliza modelos de DL para identificar e localizar objetos em imagens. Esta suporta modelos de deteção como: Faster R-CNN, SSD e R-FCN, que são pré-treinados em vastos conjuntos de dados, como o COCO (*Common Objects in Context*), permitindo uma deteção precisa em várias categorias de objetos.

Além de usar modelos pré-treinados, é possível treinar um modelo próprio usando um conjunto de dados personalizado. Com esta ferramenta também é possível fazer a avaliação do modelo.

3.3.4.2 TensorBoard

Os modelos de rede neuronal são frequentemente estruturas grandes e complicadas e podem gerar confusão. O TensorBoard é uma ferramenta de visualização que facilita a compreensão e a depuração de programas do TensorFlow. Os utilizadores podem utilizar o TensorBoard para visualizar facilmente o gráfico de cálculo de um modelo, as métricas de formação e os valores dos parâmetros. Um módulo do TensorFlow, tf.summary, cria operações para guardar a definição do gráfico e registar estatísticas de treino. Também as grava como um ficheiro de eventos no disco rígido num formato que o TensorBoard pode ler. Estes ficheiros serão visualizados num navegador quando o TensorBoard for iniciado.

3.3.5 ReLU6

O ReLU6 é uma modificação da função de ativação linear retificada original (ReLU - *Rectified Linear Unit*). A ReLU é uma função linear por partes que produz a entrada diretamente se for positiva, caso contrário, produzirá zero. Esta função tornou-se um padrão para muitos tipos de redes neuronais por ao mesmo tempo [169].Conforme ilustrada na Figura 36 e matematicamente expressa na Equação 3.

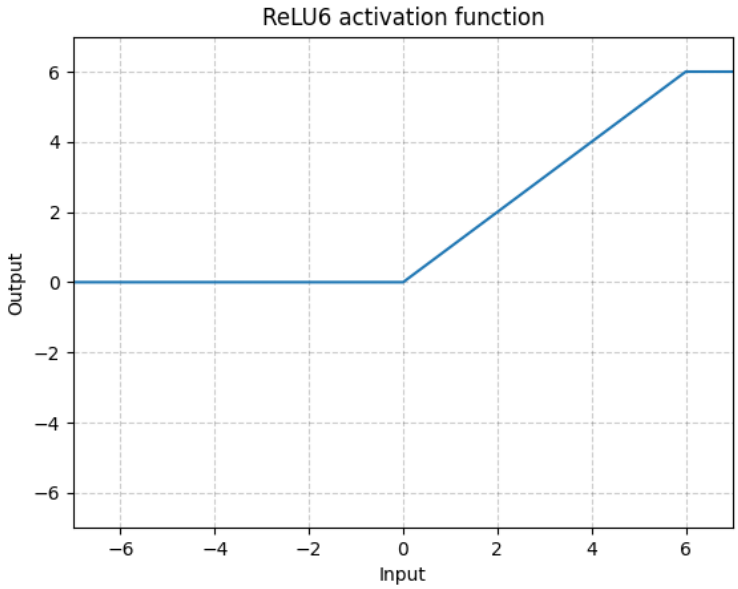


Figura 36 ReLU6 [86]

$$f(x) = \max(0, x) \qquad (3)$$

ReLU6 limita a saída da função ReLU a um tamanho máximo de 6. Isso significa que qualquer entrada abaixo de 0 será descartada, quanto qualquer entrada acima de 6 será convertida em 6 [170], matematicamente expressa na Equação 4.

$$f(x) = min\ (max\ (0, x), 6) \qquad (4)$$

ReLU6 tem diversas aplicações para aumentar a robustez de redes neuronais. Uma das aplicações mais significativas encontra-se em dispositivos móveis, onde os recursos computacionais são limitados. Conforme mencionado anteriormente, o ReLU6 pode ajudar a manter o desempenho da rede neuronal mesmo em aritmética de baixa precisão. Isto pode ser especialmente útil em dispositivos móveis onde os recursos computacionais podem ser limitados. Para esta dissertação, este conceito é importante para entender a sua função dentro do MobileNet.

3.3.6 MobileNet

Como discutido na secção 2.2.3, o MobileNet é um modelo, desenvolvida pela Google, precursor para o equilíbrio da latência e precisão. Estes modelos são baseados numa arquitetura simplificada que usa convoluções separáveis para construir uma rede neuronal convolucional (CNN – *Convolutional Neural Network*). Este tipo de modelo tem-se mostrado eficaz numa ampla gama de aplicações, como deteção e classificação de objetos.

A primeira versão do MobileNet tem duas camadas, a primeira camada é chamada de convolução profunda e realiza uma filtragem leve aplicando um único filtro convolucional por canal de entrada. A segunda camada é uma convolução 1×1, chamada convolução pontual, que é responsável pela construção de novos recursos por meio do cálculo de combinações lineares dos canais de entrada, é utilizado ReLU6 para comparação.

Para o MobileNetV2, existem dois tipos de blocos. Um é o bloco residual com passo 1, o outro é o bloco com passo 2 para redução. Existem três camadas para ambos os tipos de bloco, a primeira camada é uma convolução 1x1 com ReLU, a segunda é a convolução profunda e a terceira é outra camada 1x1, mas sem uma função para manter linearidade. A Figura 37 mostra a diferença entre MobileNetV1 e MobileNetV2.

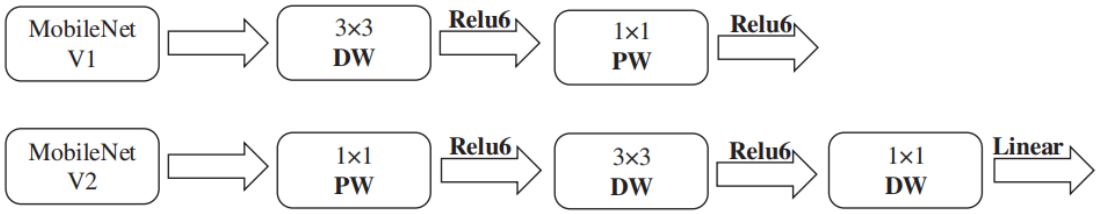


Figura 37 MobileNet vs. MobileNetV2 [171]

O estudo desenvolvido por Evan Juras [172] comparou os principais modelos de deteção de objetos ao serem aplicados a um Raspberry Pi 4. Como resultado, através desta metodologia, o modelo que obteve melhores resultados foi SSD-MobileNet-FPNLite-320×320, seguido por SSD-MobileNet-v2.

3.3.7 Roboflow

O Roboflow, que tem como logo a Figura 38, é uma plataforma de tratamento de dados para modelos de visão computacional, sendo uma alternativa a ferramentas tradicionais como o LabelImg. Diferente deste último, que se limita à anotação de dados, o Roboflow inclui funcionalidades como pré-processamento de dados, verifica a integridade dos dados e tem capacidade de exportar os dados anotados em múltiplos formatos, facilitando a interoperabilidade com diversos sistemas e modelos de treino.

Figura 38 Logo Roboflow

3.3.8 Outras bibliotecas utilizadas

Para além das bibliotecas tradicionais utilizadas para desenvolver modelos de visão computacional que foram citadas neste subcapítulo, outras bibliotecas e módulos do Python também desempenham papéis fundamentais para o funcionamento do código, destacando:

1. Os: módulo utilizado para interagir com o sistema operacional, permite executar tarefas como criar e listar diretórios;
2. Shutil: Útil para operações como copiar e remover arquivos;
3. Argparse: Permite que os utilizadores especifiquem os parâmetros que querem passar para o script Python diretamente através da linha de comando;

4. Numpy: Essencial para o processamento de dados, facilita a manipulação de vetores e matrizes;
5. time: Utilizado para medir intervalos de tempo;
6. Thread: Permite a execução de múltiplas *threads* em um programa, o que é benéfico para acelerar o processamento de imagem ao dividir tarefas em várias *threads*;
7. Google protobuf (pipeline_pb2 e text_format): módulos usados para configurar e serializar dados estruturados.

3.4 Comunicação UR3e e Raspberry Pi

A comunicação entre o UR3e e o Raspberry Pi é feita através de um cabo Ethernet. Para tal, o IP dos dois dipositivos tem de ser configurado para que ambos estejam na mesma sub-rede. Neste projeto, foi utilizada a faixa de endereço “169.254.148.X”, que faz parte da faixa de endereços conhecida como “Link-local”, essa faixa trabalha entre "169.254.0.1" a "169.254.255.254", com uma sub-rede padrão de “255.255.0.0” (/16 em notação CIDR).

3.5 Controlo do UR3e

Para controlo do UR3e, a abordagem escolhida para esta dissertação foi da utilização da biblioteca socket, uma biblioteca em Python que permite a comunicação do Raspberry Pi através do protocolo TCP/IP. Esta biblioteca facilita o envio de comandos URScript diretamente para o controlador do robô. Assim, esta abordagem permite um controlar o UR3e. A Tabela 5 específica as funções mais utilizadas desta biblioteca.

Tabela 5 Códigos URScript

Natureza do comando	Função	Operação
Movimento	movej(juntas, a, v, t, r)	Move o robô para posições de juntas especificadas.
	movel(x,y,z,rx,ry,rz, a, v, t, r)	Move o robô para uma posição linear especificada.
	movep(x,y,z,rx,ry,rz, a, v, t, r)	Movimento linear em tempo real
	movec(posição_inicial, posição_final, a, v, r, modo):	Move o robô em um arco que passa pela posição intermediária e termina na posição final.

Controlo de velocidade	speedj(velocidade_juntas, a, t)	Acelera as juntas do robô até uma velocidade especificada.
	speedl(velocidade_ferramenta, a, t)	Acelera o ferramental até uma velocidade linear especificada.
Entradas/Saídas (I/O)	set_digital_out(id, state)	Define o estado de uma saída digital.
	set_analog_out(id, value)	Define o valor de uma saída analógica.
	get_digital_in(id)	Lê o estado de uma entrada digital.
	get_analog_in(id)	Lê o valor de uma entrada analógica.
Espera e Sincronização	sleep(t)	Pausa a execução do programa por um tempo especificado
	wait	Espera até que uma condição específica seja verdadeira.

3.6 Distorção de lente radial e tangencial

As lentes das câmaras reais sofrem distorção não linear. Na prática, a distorção radial da lente faz com que as linhas retas sejam mapeadas como linhas curvas. A distorção da lente radial parece mais visível nas bordas da imagem, onde a distância radial é elevada [173], como apresentado na Figura 39.

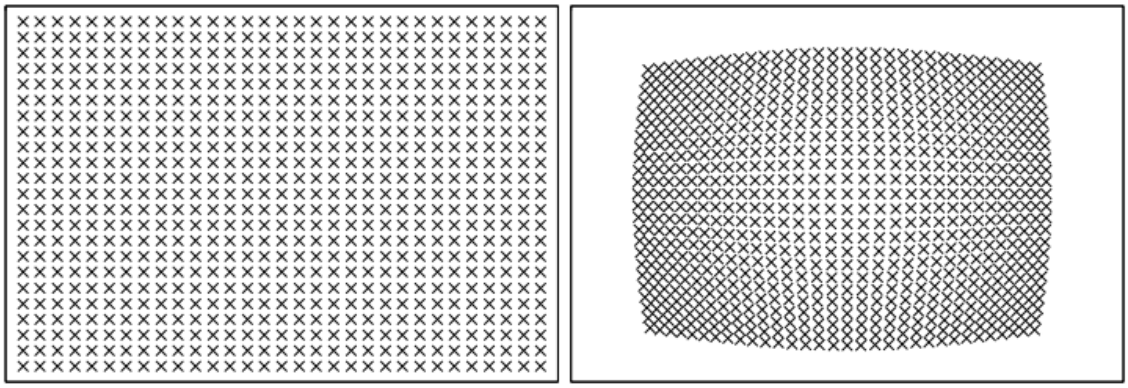


Figura 39 Malha de pixéis de uma imagem sem distorção radial e com distorção [173]

A distorção tangencial ocorre quando a lente e o plano da imagem não são paralelos. Diferentemente da distorção radial, o plano da imagem é inclinado e a distância do centro é menos importante, como exposto na Figura 40.

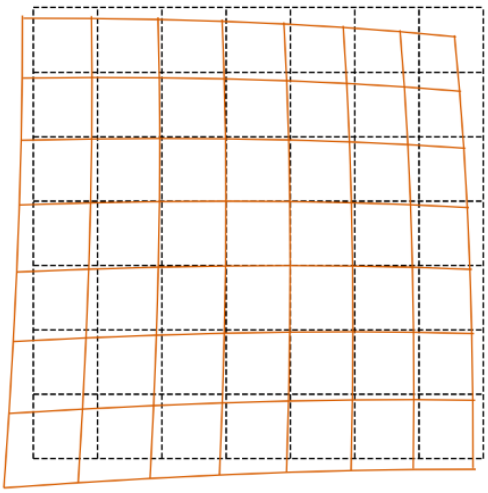
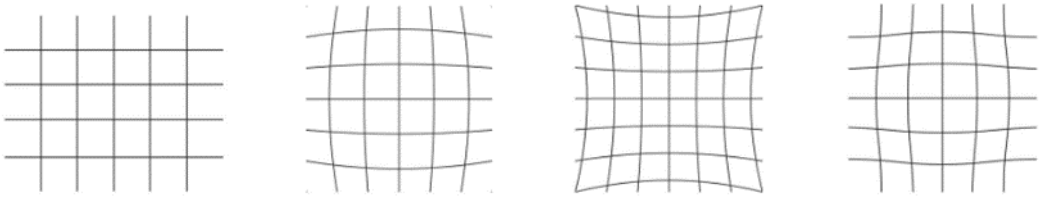


Figura 40 Distorção tangencial [174]

A distorção radial na geração de imagens ocorre devido a imperfeições no design da lente, em que os raios de luz se curvam de forma diferente a distâncias variadas do centro ótico. Esta condição faz com que as imagens se deformem. Os tipos mais comuns de distorção radial estão expostas na Figura 41.



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

Figura 41 Tipos de distorção de lente [175], Sem distorção (a), Distorção de barril (b), Distorção de almofada de alfinetes (c) e distorção de bigode (d)

Distorção de barril: Este tipo de distorção observado na Figura 41 (b) faz com que as imagens fiquem salientes em direção às bordas, fazendo com que a parte central da imagem pareça ampliada. É comumente visto em lentes grande angulares e pode fazer com que linhas retas próximas às bordas pareçam curvadas para dentro em direção ao centro.

Distorção de almofada de alfinetes: O oposto da distorção em barril, a distorção em almofada de alfinetes, tal como apresentado na Figura 41 (c), faz com que as imagens sejam comprimidas nas bordas. É mais comum em lentes telefoto, onde as linhas retas próximas às bordas da imagem aparentam curvar para fora a partir do centro.

Distorção de bigode: Um tipo complexo de distorção que combina elementos das distorções de barril e de almofada de alfinetes, e cujo resultado se pode ver na Figura 41 (d). Ela aparece como um padrão ondulado na imagem, onde as linhas se distorcem para dentro e para fora. Esse tipo é menos comum, mas pode ser particularmente difícil de corrigir.

3.6.1 Matriz da câmara

A matriz da câmara é uma representação matemática que descreve as características internas de uma câmara. Esta matriz é parametrizada por Hartley e Zisserman [176] como demostrado na Equação 5.

$$K = \begin{matrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{matrix} \qquad (5)$$

O f_x e f_y representam na Figura 42 a distância focal, que é a distância entre o orifício e o filme. A distância focal é medida em pixéis. Numa situação ideal, f_x e f_y possuem o mesmo valor, porém, fatores como falhas no sensor da câmara digital, falhas no dimensionamento no pós-processamento e distorções derivadas da lente, fazem o valor de f_x e f_y frequentemente ser um valor distinto [177].

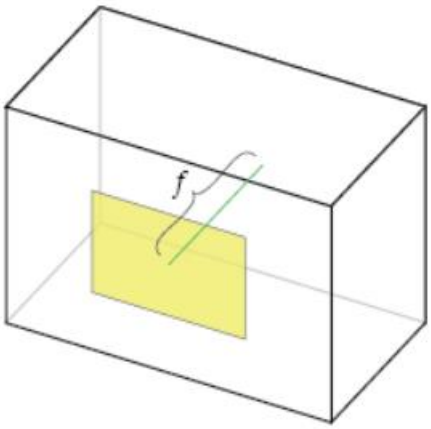


Figura 42 Distância focal [177]

O deslocamento do ponto principal é representado na Figura 43 por x_0 e y_0. O "eixo principal" da câmara é a linha perpendicular ao plano da imagem que passa pelo orifício. A sua interseção com o plano da imagem é chamada de "ponto principal" [177].

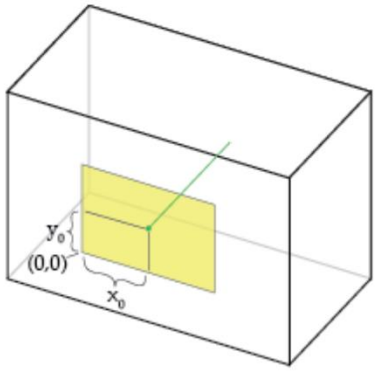


Figura 43 Representação do deslocamento do ponto principal [177]

O s representa a inclinação do eixo, que causa distorção de cisalhamento na imagem projetada, na grande maioria das vezes esse valor será 0, apenas em pouquíssimas exceções, devido ao processo de pós processamento, esse valor pode ser diferente de 0 [177].

3.6.2 Coeficiente de distorção

Os coeficientes são parâmetros que descrevem como as lentes da câmara distorcem a imagem captada. Esta distorção necessita de ser corrigida para melhorar a precisão das medidas feitas com imagens. Este coeficiente é descrito pela Equação 6.

$$\text{Coeficiente de distorção} = (\, k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3) \qquad (6)$$

Os valores de k_1 e k_2 são os coeficientes de distorção radial, p_1 e p_2 são coeficientes de distorção tangencial e k_3 também se trata de um coeficiente para distorção radial, mas para distorções radiais significativas, com distorção em níveis mais altos de π^6.

3.7 Iluminação

As imagens não podem existir sem luz. Para produzir uma imagem, a cena deve ser iluminada com uma ou mais fontes de luz. As fontes de luz geralmente podem ser divididas em fontes de luz pontuais (por exemplo, uma pequena lâmpada) e de área (por exemplo, o Sol) [178]. Uma fonte de luz pontual origina-se em um único local no espaço e possui certa intensidade e espectro de cores, ou seja, uma distribuição em comprimento de onda L (λ).

As fontes de luz de área são mais complexas. Uma fonte de luz de área simples, como uma luminária de teto fluorescente com um difusor pode ser modelada como uma área retangular finita que emite luz igualmente em todas as direções.

A seleção e o posicionamento de câmaras e fontes de luz são uma das etapas mais importantes na criação de um sistema de CV. A obtenção de imagens de alta qualidade pode simplificar muito os algoritmos de visão e melhorar sua fiabilidade [179], em aplicações de visão mecânica, uma iluminação boa e uniforme é importante. A iluminação não uniforme por uma fonte de luz externa pode causar mais danos do que benefícios e falhar no processo de segmentação e, consequentemente, no processo de deteção [180] a diferença entre uma região não iluminada e mal iluminada é demonstrada Figura 44.

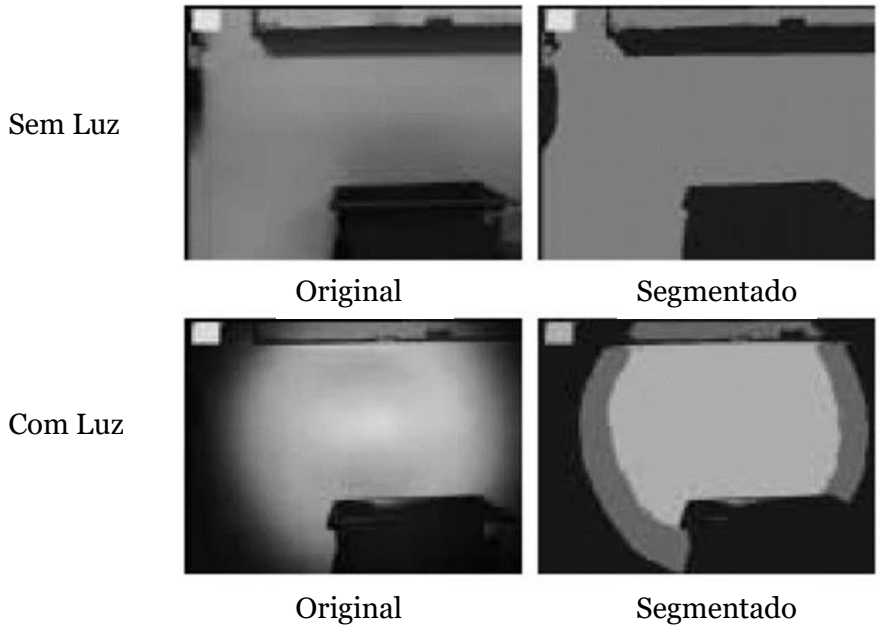


Figura 44 Efeito da Luz na segmentação [181]

Áreas bem iluminadas, de maneira uniforme, aumentam a fiabilidade da deteção num ambiente que utiliza visão computacional, e áreas má iluminadas resultam numa segmentação errada do foco de luz.

3.8 Área de trabalho

A área de trabalho foi desenvolvida de modo a garantir que as imagens obtidas possuíssem qualidade e resolução suficientes para permitir a identificação de defeitos e características únicas dos frutos. Outro requisito importante foi manter a câmara

perpendicular à área de atuação, de modo a facilitar a conversão da distância em pixéis para a distância em metros. Além disso, foi essencial uma configuração que proporcionasse uma boa mobilidade ao braço robótico em todas as fases do trabalho. A Figura 45 ilustra a montagem da área de trabalho:

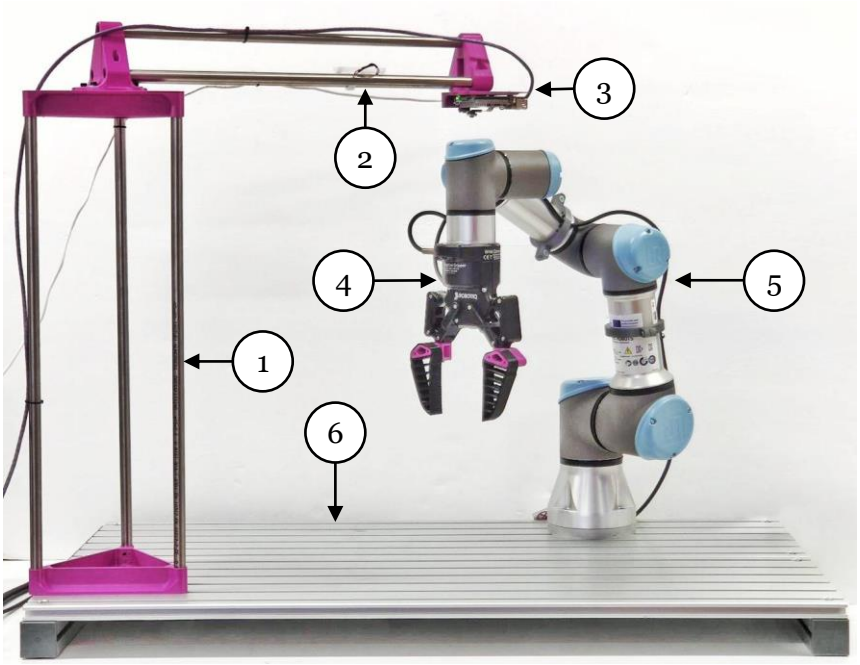


Figura 45 Área de trabalho

Nesta área de trabalho são identificadas várias componentes, sendo:

1. Torre: Esta torre foi desenvolvida com impressão 3D e tubos de inox a fim de ser o suporte para os Equipamentos 2 e 3, cumprindo o requisito de ser a 180 graus do Equipamento 6.
2. Fonte de Luz: Dois leds de 12V ligados em série com a função de ser uma fonte de luz pontual que atua em direção ao Equipamento 6.
3. Raspberry Pi 5 com PiCamera módulo 3: Com função de receber as imagens e enviar comandos ao Equipamento 5.
4. Atuador Robotiq 2F-85: Garra robótica, responsável por parte da manipulação dos frutos.
5. UR3e: Braço Robótico que recebe as informações do Equipamento 3, responsável por parte da manipulação dos frutos.
6. Base de fixação do sistema: mesa de alumínio em perfil minitec, responsável pela fixação de todo o sistema.

3.9 Calibração da câmara

De modo a eliminar as distorções descritas no subcapítulo 3.6, é necessária uma calibração da câmara, para isto, foi criado um tabuleiro de xadrez utilizando o código do Anexo A.5 (ver Figura 46).

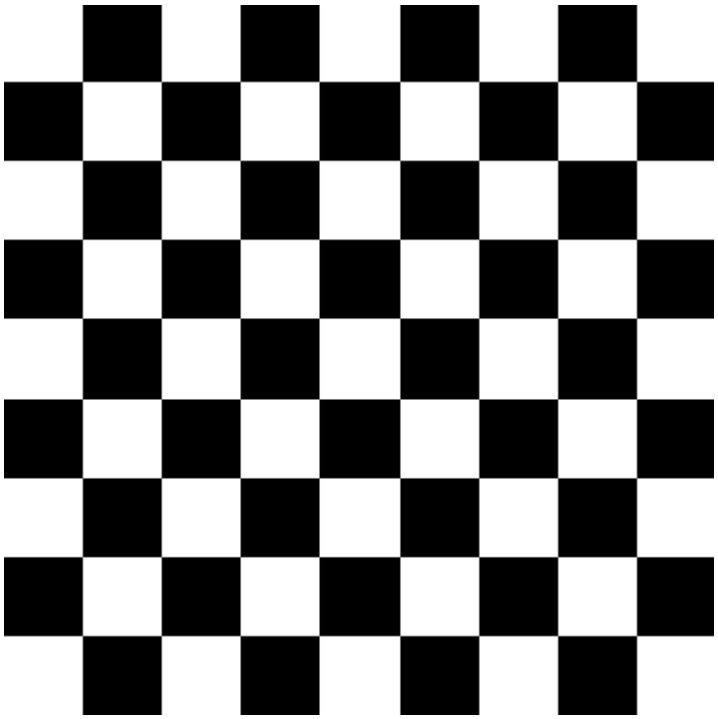


Figura 46 Tabuleiro de xadrez

O Tabuleiro é então impresso em uma folha A4 e colocado na superfície da área de trabalho (ver Figura 45). Foram adquiridas 50 fotos de diversos ângulos e distâncias do tabuleiro, de modo que cobrisse uma ampla gama de perspetivas. É então utilizado o código do Anexo X que utiliza a função cv2.findChessboardCorners, para encontrar o canto do tabuleiro de xadrez nas fotos (ver Figura 47) e depois armazenado numa lista. Por fim, utiliza-se a função cv2.calibrateCamera para calcular a matriz da câmara e o coeficiente de distorção. O código utilizado para tirar fotos e calibrar está no Anexo A.3 e A.4.

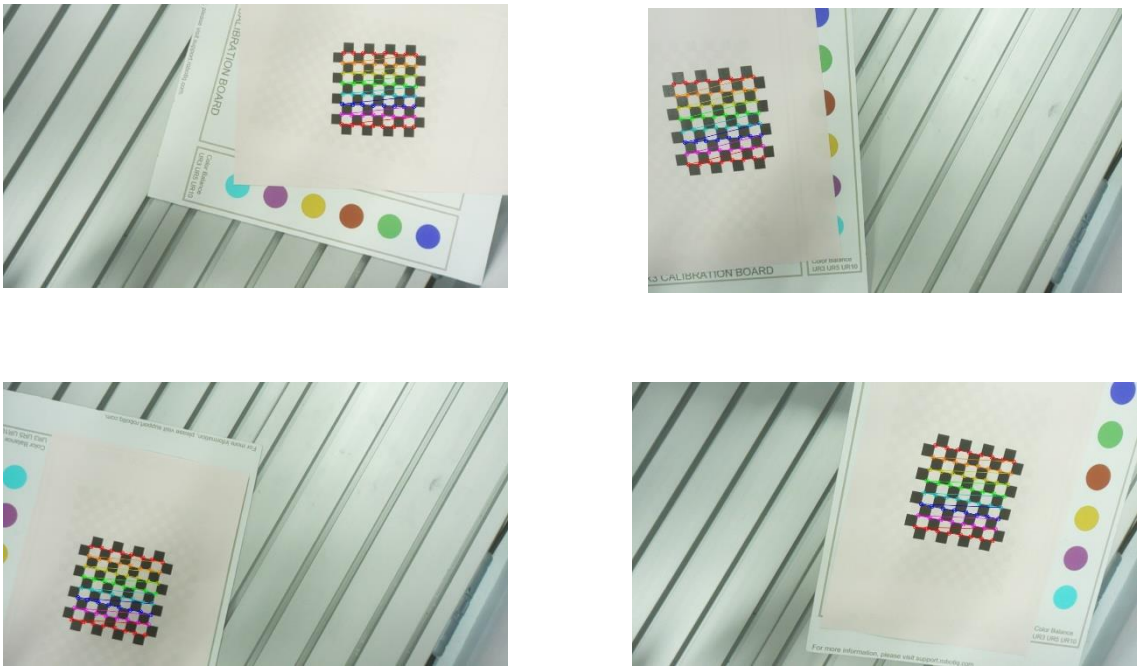
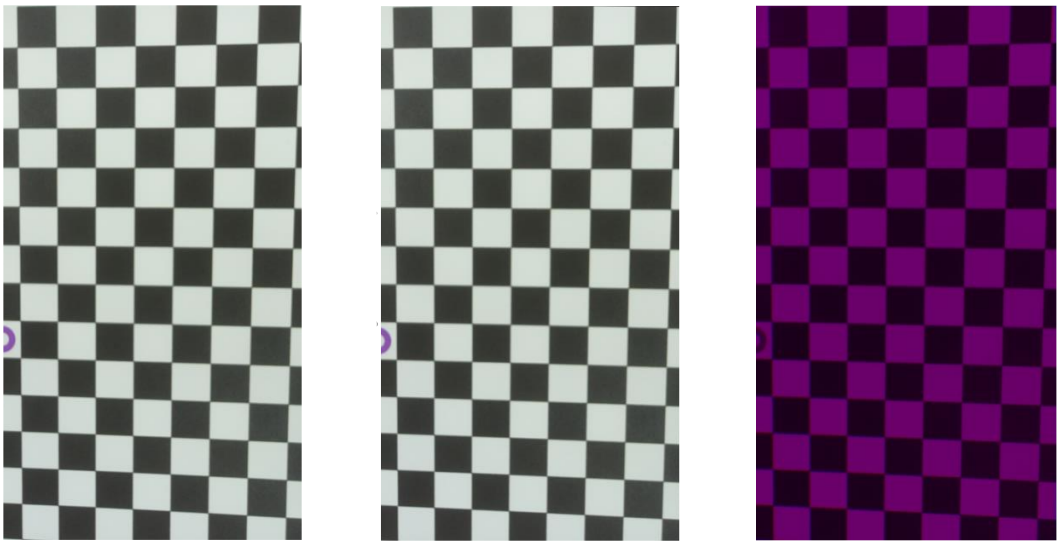


Figura 47 Exemplos do resultado da utilização do cv2.findChessboardCorners

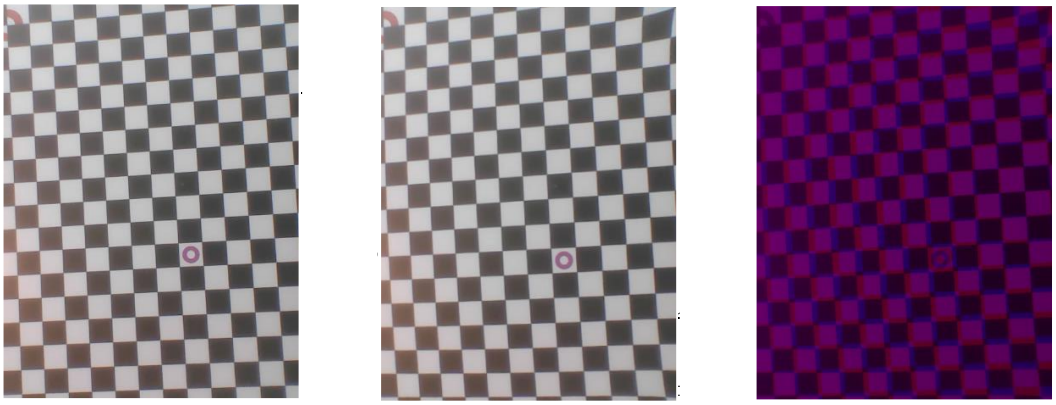
A PiCamera módulo 3 possui uma baixa distorção, e por isso a calibração não é facilmente percetível. Para efetuar a comparação, as imagens foram sobrepostas com transparência, sendo uma colorida de azul e outra colorida de vermelho. Os resultados obtidos na Figura 48 e o resultado do coeficiente de distorção: [-0.08039693, 0.66372094, 0.00554512, 0.00156936, -1.30104356] confirmam que a distorção da câmara realmente é muito reduzida.



(a) (b) (c)

Figura 48 Comparação de uma imagem capturada pela PiCamera módulo 3 pré calibração (a), pós calibração (b) e sobreposição entre as duas (c).

Porém, em algumas câmaras este processo deve ser realizado. A Figura 49 mostra esse mesmo processo com uma câmara 04081-00211500 - HD 3.3V.



(a) (b) (c)

Figura 49 Comparação de uma imagem capturada pela câmara 04081-00211500 - HD 3.3V, pré calibração (a), pós calibração (b) e sobreposição entre as duas (c).

Na Figura 49 demostra a diferença da imagem distorcida e sem distorção, a Figura 49 (c) apresenta o contraste das duas sobrepostas, deste modo, é possível perceber a diferença.

3.10 Critérios gerais de avaliação de um modelo

Existe diversas maneiras de fazer a avaliação de um modelo computacional, utilizando-se indicadores específicos, para a deteção de objetos, sendo os mais utilizados:

Precisão: é uma medida de desempenho que quantifica a exatidão de um modelo na realização de previsões positivas. É definida pela equação 7.

$$Precisão = \frac{Positivo\ Verdadeiro}{Positivo\ Verdadeiro + Positivo\ Falso} \qquad (7)$$

Recuperação: A importância dessa métrica reside na sua capacidade de medir a capacidade do modelo para detetar todos os casos positivos, o que o torna uma métrica crítica em situações em que a falta de instâncias positivas pode ter consequências significativas. A recuperação quantifica a proporção de instâncias positivas que o modelo identificou com êxito. Isto fornece informações sobre a eficácia do modelo na captura do conjunto completo de objetos ou características relevantes nas imagens analisadas, pode ser calculado pela equação 8.

$$Recuperação = \frac{Positivo\ verdadeiro}{Positivo\ verdadeiro + Negativo\ falso} \qquad (8)$$

Pontuação F1: A pontuação F1 é uma métrica de desempenho que combina a precisão e a recuperação num único valor, fornecendo uma medida equilibrada do desempenho de um modelo de CV. É definida como a média harmónica de precisão e recuperação e pode ser calculada pela equação 9.

$$Pontuação\ F1 = \frac{2 * (\ precisão * recuperação)}{precisão + recuperação} \qquad (9)$$

A importância da pontuação F1 decorre da sua utilidade em cenários com distribuições de classes desiguais ou quando os falsos positivos e os falsos negativos têm custos diferentes. Ao considerar tanto a precisão (a exatidão das previsões positivas) como a recuperação (a capacidade de identificar todas as instâncias positivas), a pontuação F1 oferece uma avaliação abrangente do desempenho de um modelo, particularmente quando o equilíbrio entre falsos positivos e falsos negativos é crucial.

Exatidão: Esta métrica mede a percentagem de instâncias que o modelo classificou corretamente, considerando tanto de classes positivas como negativas, e pode ser calculada pela equação 10.

$$Exatidão = \frac{Positivo\ verdadeiro + Negativo\ verdadeiro}{Positivo\ verdadeiro + Positivo\ falso + Negativo\ verdadeiro + Negativo\ falso} \qquad (10)$$

A importância da exatidão decorre da sua capacidade de fornecer uma medida direta do desempenho global do modelo. Dá uma ideia geral do desempenho do modelo numa determinada tarefa, como a deteção de objetos, a classificação de imagens ou a segmentação. No entanto, a precisão pode não ser adequada em situações com desequilíbrios de classe significativos, uma vez que pode dar uma impressão enganadora do desempenho do modelo. Nesses casos, o modelo pode ter um bom desempenho na classe maioritária, mas um desempenho fraco na classe minoritária, o que leva a uma precisão elevada que não reflete com exatidão a eficácia do modelo na identificação de todas as classes.

Intersecção sobre União (IoU): também conhecida como índice de Jaccard, é uma métrica de desempenho normalmente utilizada na avaliação de modelos de visão por computador. É particularmente importante para tarefas de deteção e localização de objetos. O IoU é definido como o rácio entre a área de sobreposição entre a caixa delimitadora prevista e a caixa delimitadora da verdade terrestre e a área da sua união. Em termos simples, o IoU mede o grau de sobreposição entre a previsão do modelo e o alvo real, expresso como um valor entre 0 e 1, sendo que 0 indica que não há sobreposição e 1 representa uma correspondência perfeita. Ele pode ser calculado pela equação 11.

$$IoU = \frac{Área\ de\ Interseção}{Área\ de\ União} \qquad (11)$$

3.11 Método de treino do modelo

A Figura 50 expõe o fluxo de trabalho de treino do modelo de deteção de objetos, a criação de anotações, divisão em treino, teste e validação e todo o trabalho de pré-processamento (auto-orientação e redimensionamento) é realizado no Roboflow, que pode gerar os arquivos em formato tfrecord.

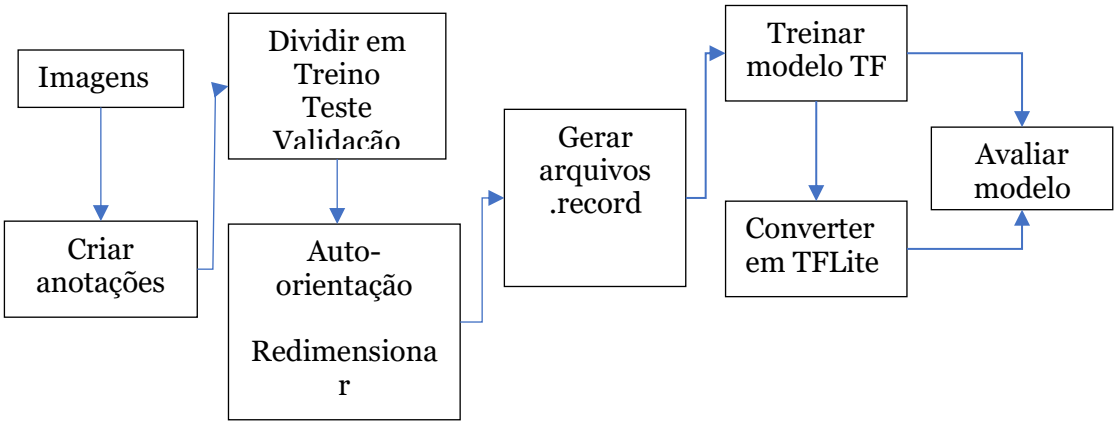


Figura 50 Fluxo de trabalho para treinar o modelo

O treino do modelo de CV foi realizado utilizando o TensorFlow Object Detection API com o modelo pré-treinado ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8. Este modelo, que utiliza SSD com Mobilenet v2 e FPN-lite como extrator de características, é uma adaptação móvel eficiente, ideal para dispositivos com recursos computacionais limitados, como o Raspberry Pi 5.

O modelo foi treinado por 40.000 iterações com um tamanho de lote de 16 imagens por etapa de treino. A configuração para treino numa TPU-8 visou atingir uma precisão média (mAP) de 0,22 na validação do COCO17, o que exigiu ajustes importantes:

1. Definição do número de classes de objetos a serem detetadas.
2. Uso de IoU para medir a precisão entre as caixas previstas e os objetos reais, com um limiar de correspondência de 0,5.
3. Produção de âncoras em diversas escalas e proporções para capturar objetos de tamanhos e formas variados.
4. Redimensionamento das imagens para 320x320 pixéis para padronizar o tamanho de entrada do modelo.
5. Aplicação de um preditor de caixa convolucional, que é ajustado durante o treino para melhorar a deteção.
6. Implementação de uma função de perda que combina perda de classificação focal sigmoide ponderada com perda de localização L1 suave, otimizando a deteção e localização precisas.

Após o treino, o modelo foi convertido para o formato TensorFlow Lite, facilitando a implementação em dispositivos como o Raspberry Pi 5 com desempenho satisfatório, ambos os modelos (antes e depois da conversão) serão avaliados.

3.11.1 Dados utilizados

Para esta dissertação, apesar de ter sido desenvolvido um mecanismo de técnicas de aquisição de dados, não foi criada uma base de dados. Foram utilizadas bases de dados disponíveis como públicas.

Foram utilizadas três bases de dados, duas para laranjas e outra para tomates. As bases de dados das laranjas foram divididas em duas: A primeira base de dados, com imagens sem anotações, faz parte do estudo realizado por Cristovão [182]. Este estudo dividiu a base de dados em quatro grupos principais, laranjas frescas, com manchas pretas, com cancro e com moagem. Esta base de dados inclui 686 imagens, acompanhadas por 706 etiquetas correspondentes a quatro classes distintas. A Figura 51 demostra um exemplo de cada classe. O equilíbrio entre as classes, a distribuição de tamanho das imagens e o mapa de calor das anotações dentro da imagem são ilustrados nas Figuras 52 a 54.

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

Figura 51 Exemplos de amostras da base de dado laranja, com representação das quatro classes: moagem (a) cancro (b) fresca (c) e mancha preta (d)

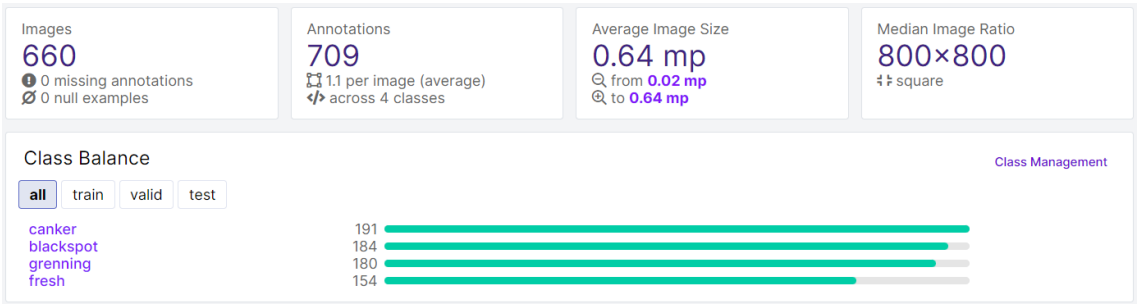


Figura 52 Base de dados de laranjas - Quatro classes

Figura 53 Informações sobre a dimensão – Base de dados de laranjas - Quatro classes

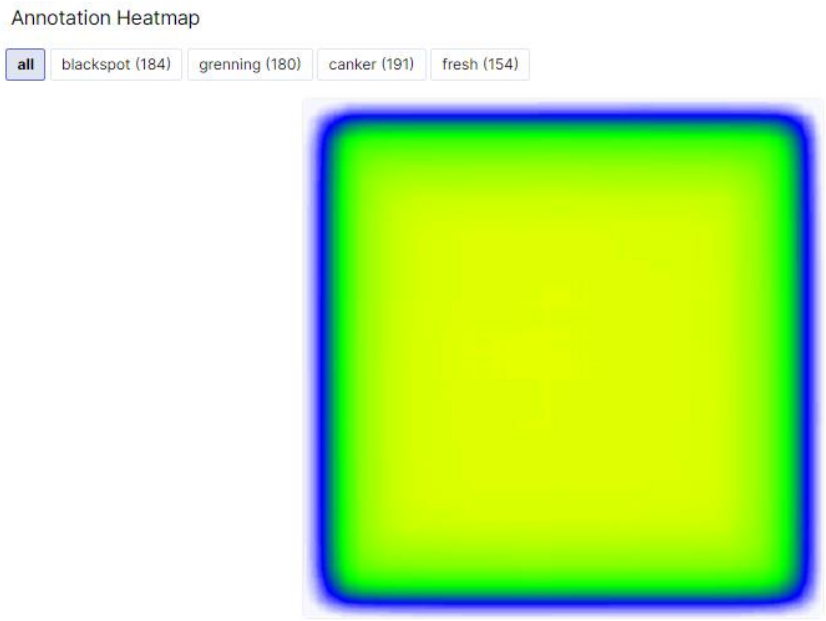


Figura 54 Mapa de calor de anotações geral - Base de dados de laranjas - Quatro classes

A segunda base de dados de laranjas é composta com imagens publicas disponíveis no Kaggle [183] e Roboflow Universe [184], composta de 5058 imagens, com 13396 anotações para duas classes, o equilíbrio de classe é demostrados na Figura 55, a distribuição de tamanho e o mapa de calor das anotações dentro da imagem são ilustrados nas Figuras 56 a 58.

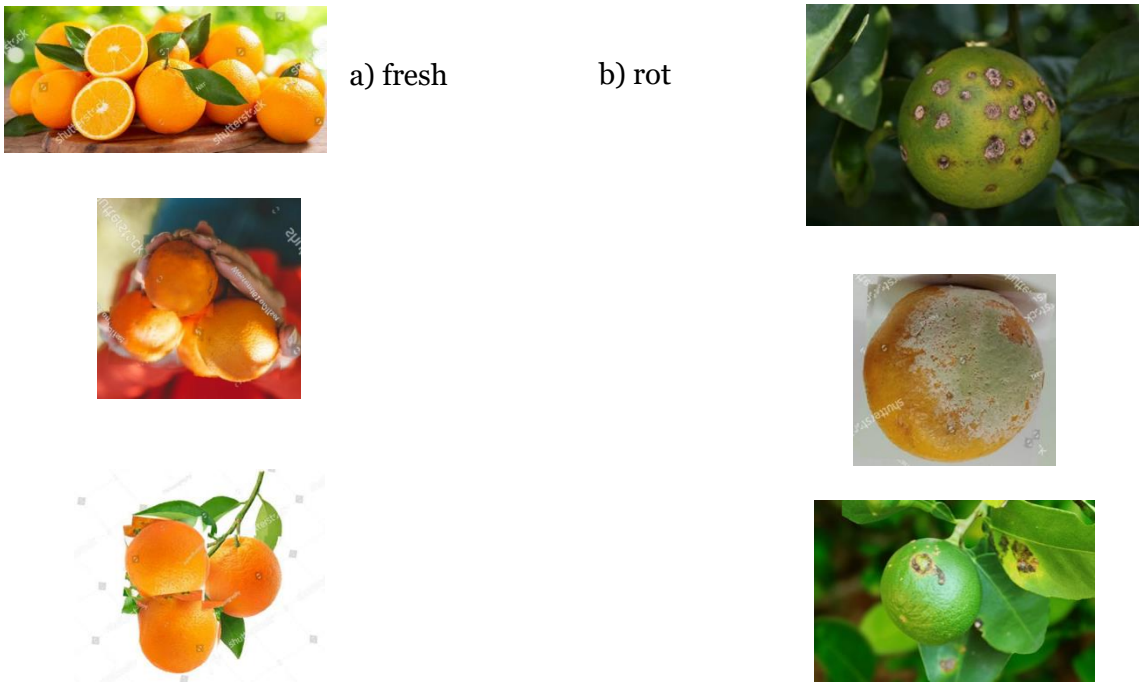


Figura 55 Amostras da base de dados de laranjas com duas classes

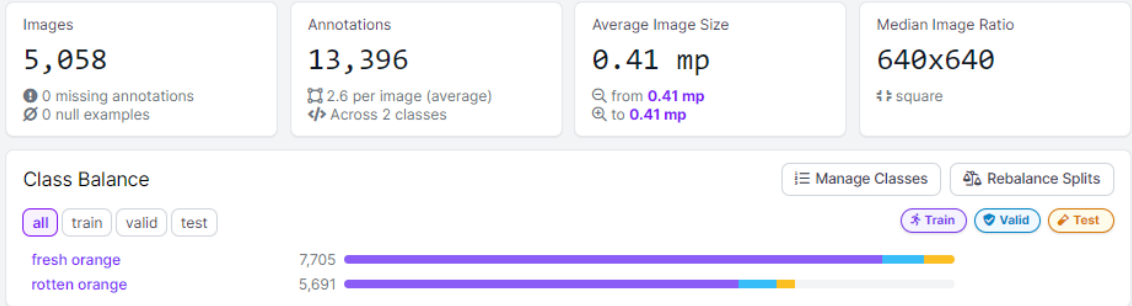


Figura 56 Equilíbrio de classes – Base de dados de Laranjas - Duas classes

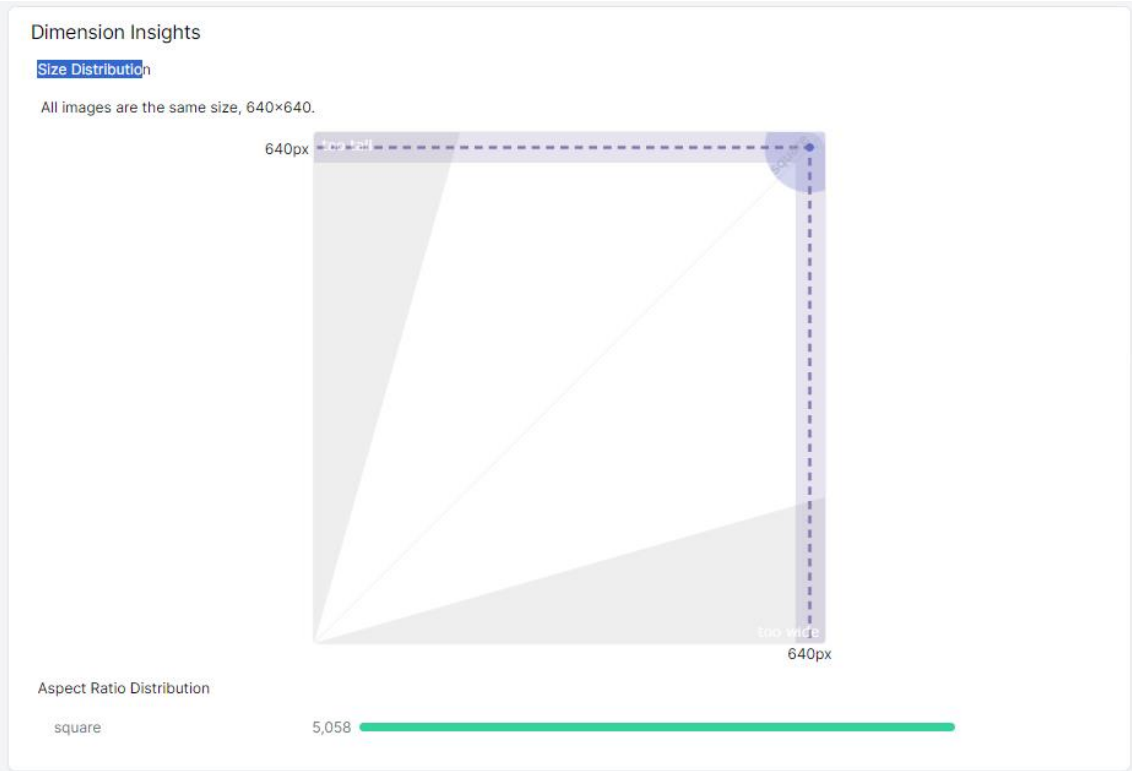


Figura 57 Informações sobre a dimensão – Base de dados de laranjas – Duas classes

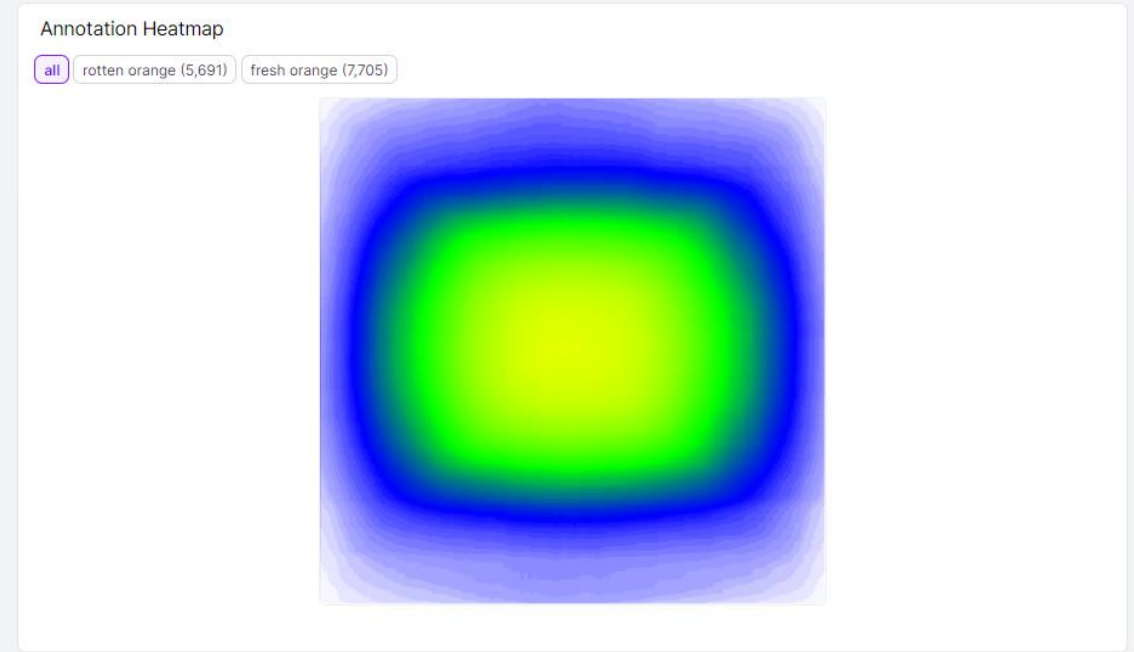


Figura 58 Mapa de calor de anotações geral -Base de dados de laranjas – Duas classes

Para tomate, foram utilizadas 7454 imagens com 55071 anotações, para esta base de dados, foram feitas duas distribuições de classe. A primeira distribuição tem anotações para quatro classes, tomate maduro, sobre maduro, não maduro e apodrecido. A segunda distribuição tem anotações para duas classes, agrupando tomate maduro e não maduro numa classe de tomate saudável, e agrupando tomate sobre maduro e apodrecido numa classe de tomate não saudável. A Figura 59 demostra exemplos de amostra dessa base de dados.

Figura 59 Amostras base de dados de tomate – Quatro classes

Os dados foram retirados de base de dados públicas disponíveis no Kaggle e Roboflow Universe. O equilíbrio entre as classes, o tamanho médio das imagens e a distribuição dos tamanhos são ilustrados nas figuras 60 a 62.

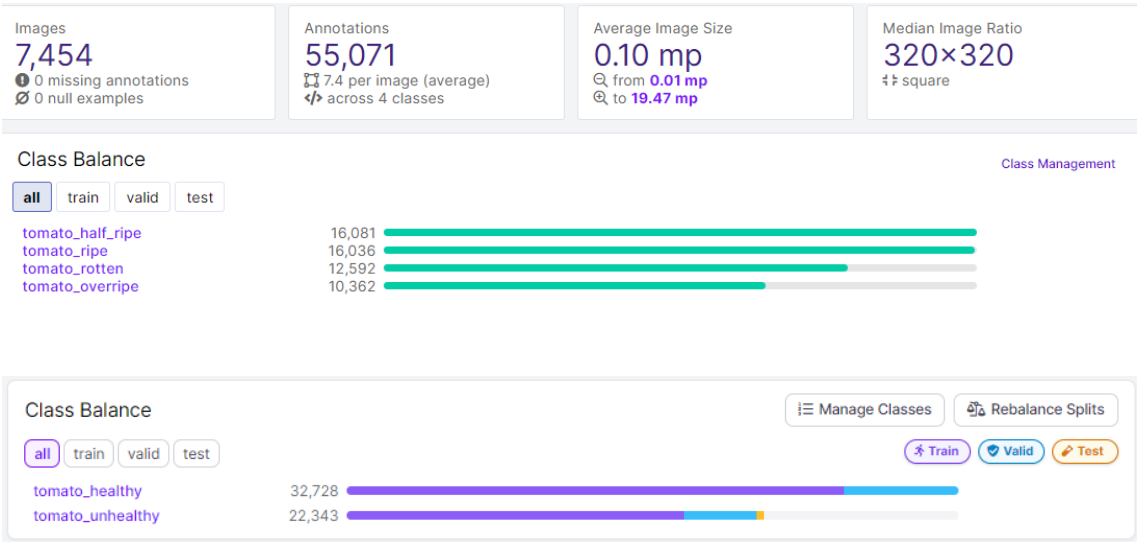


Figura 60 Equilíbrio de classes – Base de dados de Tomate – Quatro classes e Duas Classes

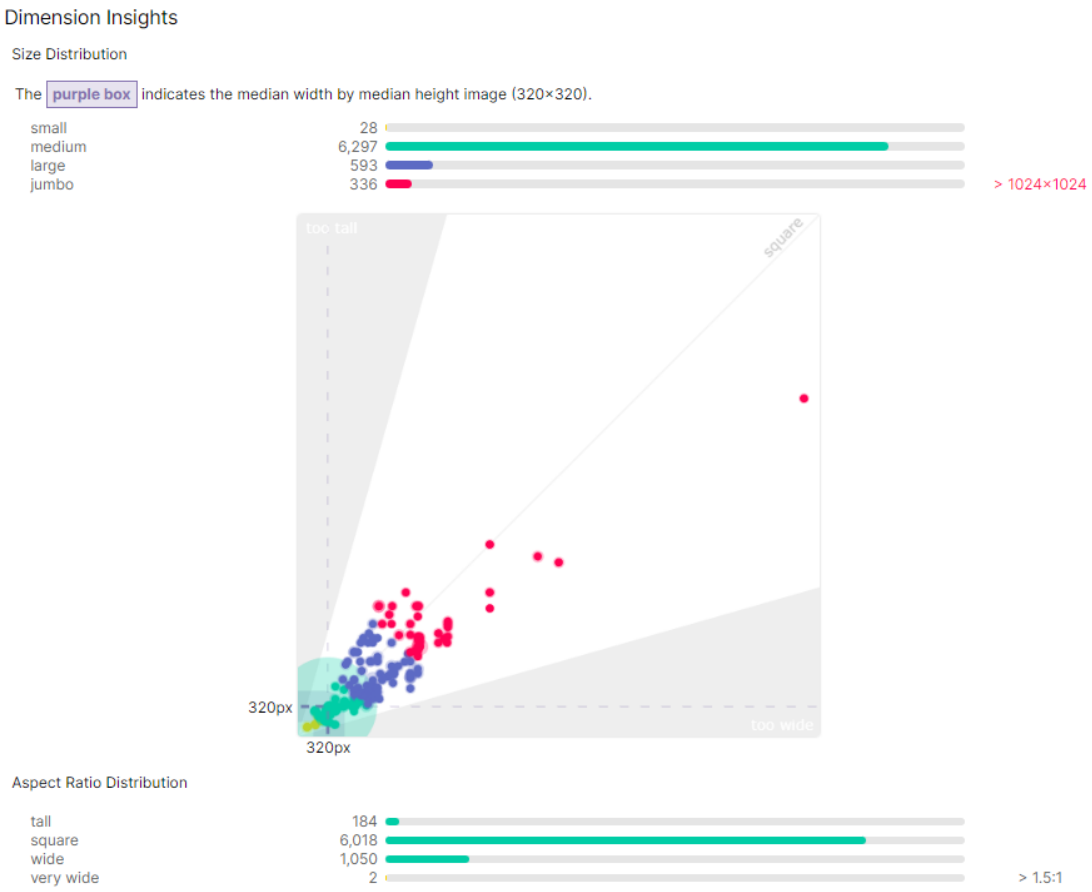


Figura 61 Informações sobre a dimensão – Base de dados de Tomate

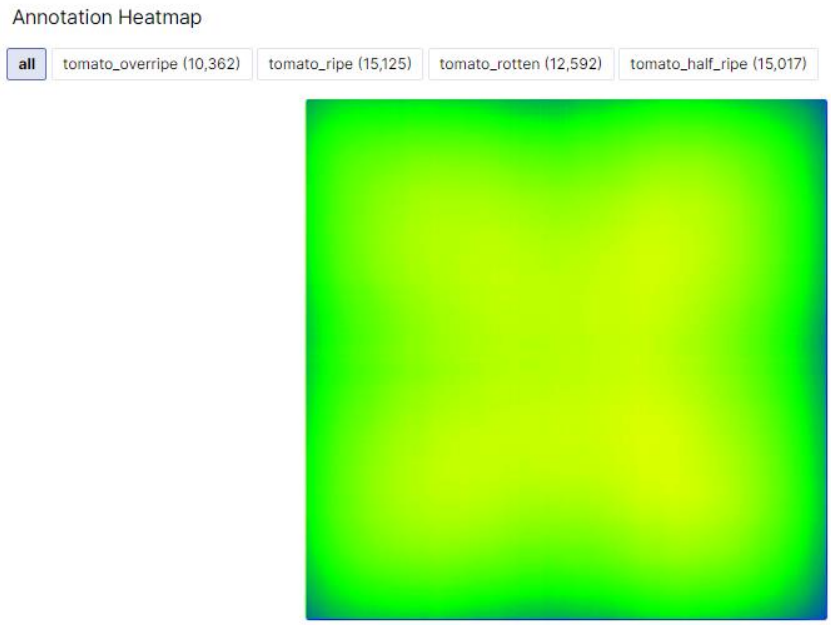


Figura 62 Mapa de calor de anotações geral -
Base de dados de Tomate – Quatro classes

3.11.2 Avaliação do modelo

O modelo será avaliado em dois momentos: antes e depois da conversão para TensorFlow Lite. Para a avaliação inicial, é utilizada a ferramenta TensorFlow Object Detection API, que possui configurações específicas para avaliar modelos usando os dados de treino. Esta ferramenta fornece informações sobre a precisão média (AP) e a recuperação média (AR), dividindo os resultados por faixa de IoU e tamanho do objeto detetado em pixéis. Após a conversão, não é possível utilizar o TensorFlow Object Detection API. Portanto, é utilizada a ferramenta desenvolvida por Cartucho [185], que fornece informações da AP e AR, dividindo por faixa de IoU.

3.12 Lógica do código

Um diagrama de blocos com uma visão geral do código é demostrado na Figura 63.

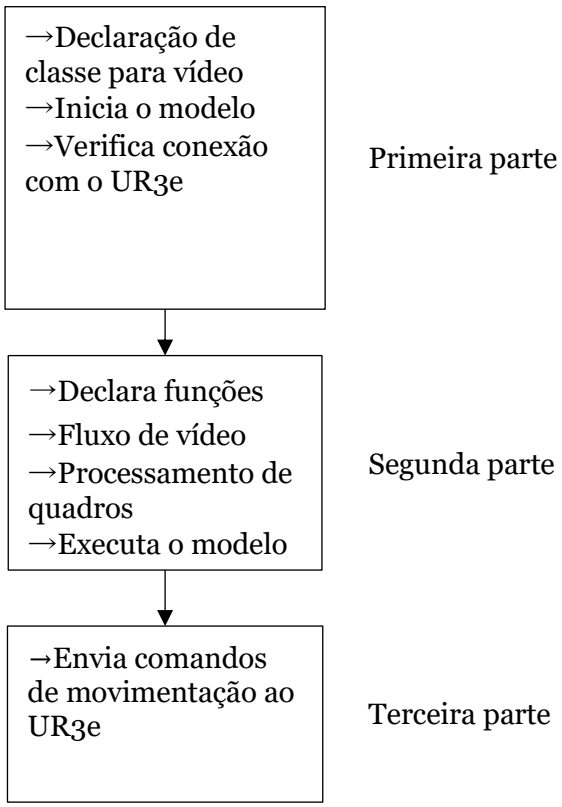


Figura 63 Diagrama com a visão geral do código

Na primeira parte do código, é declarada uma classe "VideoStream", onde na inicialização é escolhida a resolução que será utilizada, o tamanho da imagem, e realiza todos os procedimentos necessários para ligar a picamera. A seguir são utilizados quatro métodos: *start, update, read e stop*. Todos são utilizados para controlo da câmara.

Em seguida, são definidos todos os métodos de ativação do modelo em TensorFlow Lite, definindo o caminho até o modelo, carregando os mapeamentos e lista de rótulos, e inicia-se o interpretador do TensorFlow Lite para alocar os tensores necessários para a deteção. Recuperam-se os detalhes do modelo, além de configurar os parâmetros de normalização para os dados de entrada, é também nesta parte que se testa a conexão com o UR3e.

A segunda parte do código declara as funções (Tabela 6) que são utilizadas na terceira parte do código. Esta também trata do *loop* de deteção, inicia o fluxo de vídeo e faz os processamentos de quadros. Este processamento envolve converter de BGR para RGB, redimensionar os dados, normalizar pontos flutuantes. Esta parte existe de forma a adequar os dados de entrada ao formato esperado pelo modelo. O modelo é executado com o quadro de entrada e os resultados da deteção, incluindo caixas delimitadores, índices de classe e pontuações de confiança são recuperados. Deste modo, é possível calcular as coordenadas da caixa delimitadora e ajustando-a dentro do quadro, uma caixa

e um ponto central, este ponto central será a altura e largura da caixa dividido por 2, este pontos são desenhadas ao redor do objeto detetado.

Tabela 6 Funções do código.

Função	Definição	Input	Output
pixel_para_metros	Converte coordenadas de pixel para metros.	x, y (coordenadas de pixel), resolution (resolução da câmara), fov_horizontal, fov_vertical, distancia_z (opcional)	dist_x_metros, dist_y_metros (deslocamento em metros no plano X e Y)
enviar_comando_movimento_linear	Envia um comando de movimento linear ao robô especificando posição e orientação.	x, y, z (coordenadas de posição), rx, ry, rz (orientação), v (velocidade), a (aceleração), r (raio)	N/A (comando enviado ao robô)
enviar_comando_movimento_juntas	Envia um comando para mover as juntas do robô para uma posição e orientação específicas.	x, y, z (coordenadas de posição), rx, ry, rz (orientação), v (velocidade), a (aceleração), r (raio)	N/A (comando enviado ao robô)
ler_posicao_atual	Obtém a posição atual do robô em termos de suas coordenadas e orientação.	N/A	pos (posição atual do robô)

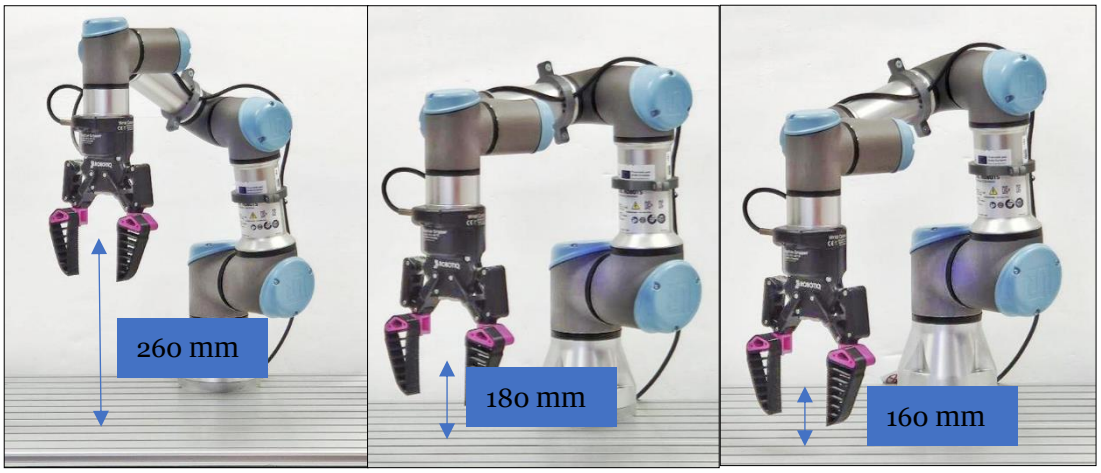
A terceira parte do código envolve o envio de comandos ao UR3e. Com auxílio das funções declaradas na parte dois, são enviados e desenvolvidos os comandos de movimentação do UR3e.

3.12.1 Definição das posições bases e lógicas de movimentação

Neste código, existem posições que sempre serão constantes, são estas posições pré-definidas e que não são atualizadas de maneira variável. Para essas posições, é utilizado

o código movej (Tabela 5), para garantir maior precisão do movimento. Estas são a posição de observação, a posição (0,0) e a posição de descarga.

Nas posições variáveis, atualizam-se apenas os valores de X, Y e Z. Os valores de RX, RY e RZ são constantes. A lógica de atualização de X e Y é realizada calculando a distância em metros do ponto (0,0) com auxílio da função pixel_para_metros. Z tem posições pré-definidas, ilustradas na Figura 64.



(a) (b) (c)

Figura 64 Posições de Z para UR3e, posição de movimentação (a), posição de aproximação (b), posição de pegar a fruta (c).

Na Figura 64 (a) pode-se observar Z3, a posição mais distante da área de trabalho de Z, utilizada para movimentações mais longas e seguras, na Figura 64 (b) observa-se Z2, a posição de aproximação da fruta e na Figura 64 (c) observa-se Z1, a posição de pegar a fruta.

3.12.2 Envio de comandos de movimentação

O envio de comandos de movimentação ao UR3e acontece quando uma fruta com defeito é detetada. Esta inicia a lógica padrão de enviar o braço para recolher o fruto defeituoso, que se define nos seguintes passos.

1. O braço é enviado à posição de reconhecimento.
2. Ao detetar uma fruta defeituosa na área de atuação, o braço é enviado ao ponto de referência (0,0), esta posição pode ser definida na função pixel_para_metro.
3. O braço então inicia movimento até a posição da fruta, atualizando X e Y, a uma distância Z3, RX, RY e RZ mantém-se constantes.

4. Ao chegar à posição da laranja, o braço vai a Z2 e logo após, em uma velocidade reduzida, a Z1. Finalizando com o fecho da garra.
5. Ao fechar a garra, o braço volta a Z2, seguindo posteriormente para a posição de descarga da fruta.
6. Ao chegar à posição de descarga, a garra abre-se e o braço volta para a posição de reconhecimento.

Se mais de uma fruta for detetada com defeito, o código ira correr novamente até retirar todas as frutas com falha. O código completo pode ser encontrado no anexo A.1.

3.13 Avaliação do sistema

Para avaliar a precisão do sistema e criar uma área de trabalho fiável para o UR3e, foi realizada uma avaliação geral do sistema de controlo em conjunto com a CV. Este teste consiste em dividir a imagem obtida pelo Raspberry Pi numa matriz de 10x10. São excluídas as linhas e colunas marginais, deixando uma matriz central de 8x8 onde os ensaios são realizados. Esta configuração pode ser observada na Figura 65.

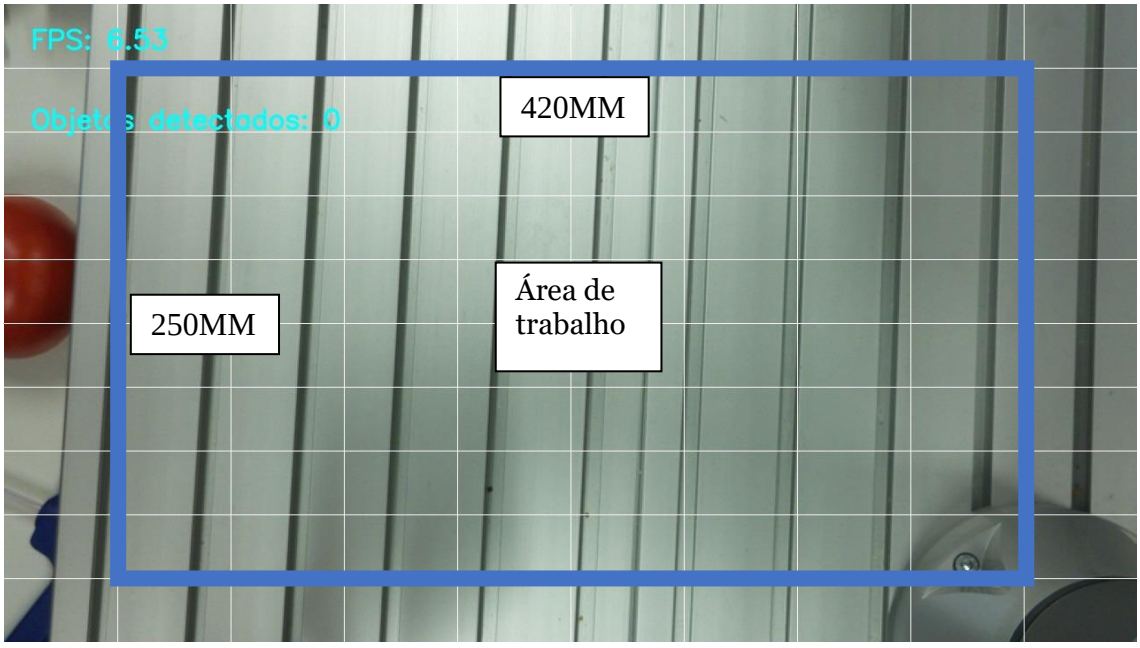


Figura 65 Imagem recebida pelo RaspBerry Pi com destaque para a área de ensaios

Em seguida, é testada a capacidade do sistema de pegar na fruta cujo modelo apresenta os melhores resultados, com o centro detetado dentro de uma célula. Cada célula será testada 10 vezes para garantir a criação de uma área de atuação fiável para o sistema.

3.14 Nota Conclusiva

Neste capítulo foram detalhados os materiais e métodos utilizados para o desenvolvimento de um sistema de controlo de qualidade, utilizando visão computacional através de Raspberry Pi 5 e uma PiCamera módulo 3 que comanda um braço robótico UR3e.

Foram expostas as bases de cada tecnologia, tanto de *hardware* como *software* e sua integração.

Relativamente ao *hardware,* foram explicados os fundamentos do funcionamento de um braço robótico, através da aplicação da convenção Denavit-Hartenberg e do conceito de cinemática inversa. Foi introduzido o Raspberry Pi 5 e a PiCamera módulo 3, com os benefícios tecnológicos e económicos de se utilizar este microcomputador, assim como a maneira de comunicar um UR3e e um Raspberry Pi 5. Foi também apresentado os efeitos da distorção na câmara e os métodos de calibração deste equipamento.

Relativamente ao *software,* foram expostas todas as tecnologias e ferramentas utilizadas para auxílio e desenvolvimento da CV, foram demostrados os parâmetros utilizados para treino do modelo e critérios de avaliação do mesmo. Foi também apresentada neste capítulo, a lógica completa do código, desde a integração do modelo de CV como o envio de comandos do Raspberry Pi 5 ao UR3e.

Os métodos adotados permitem a integração entre *hardware* e *software*, essencial para a execução da tarefa proposta.

Capítulo 4

Análise e Discussão de Resultados

Neste capítulo são apresentados e discutidos os resultados obtidos pelo treino dos modelos de deteção de objetos, são eles: as perdas de classificação, localização, regularização total, a taxa de aprendizagem e os dados referentes à precisão e recuperação baseado no IoU e tamanho da imagem. São analisadas as implicações práticas dos gráficos e tabelas e se este modelo pode ou não ser útil numa situação real. É demostrada a utilização de cada um dos quatro modelos em imagens que não possuem a mesma natureza da base de dados e examinado se o resultado condiz com os valores obtidos pelos métodos de avaliação utilizados.

Também são apresentados os dados de precisão baseados na classe após a conversão para modelo TensorFlow Lite. Também são apresentados os resultados de 640 tentativas de manipulação de um fruto usando o sistema completo. É proposto neste capítulo a criação de uma nuvem de trabalho segura para o UR3e com base nos resultados do sistema completo obtidos no capítulo anterior.

4.1 Treino dos modelos

Os modelos foram treinados utilizando a ferramenta TensorFlow Object Detection API, conforme a metodologia apresentada no subcapítulo 3.8. Os dados utilizados para o treino estão descritos na secção 3.11.1. Para a avaliação do modelo, também foi empregue o TensorFlow Object Detection API, juntamente com o TensorBoard, proporcionando uma melhor visualização das perdas ao longo de todos os passos do treino. Nos resultados, o tamanho das imagens foi classificado em pequenas, médias e grandes, conforme os seguintes critérios:

- Imagens pequenas: área menor que 32^2 pixéis
- Imagens médias: área entre 32^2 e 96^2 pixéis
- Imagens grandes: área maior que 96^2 pixéis

4.1.1 Modelo de deteção e classificação de Laranjas

Os modelos para detetar laranjas foi dividido em dois principais grupos, um com duas classes, utilizando uma base de dados públicas retiradas do Kaggle e Roboflow Universe,

e o outro com 4 classes, de uma base de dados feita para um estudo científico. Estas duas bases de dados foram descritas e analisadas na secção 3.11.1 desta dissertação.

4.1.1.1 Modelo de deteção e classificação de Laranja de duas classes com base de dados públicas

As Figuras 66 a 70 foram obtidas a partir de dados obtidos pelo TensorBoard, enquanto as Figuras 66 a 69 referem-se às perdas durante o treino do modelo TensorFlow e a Figura 70 refere-se à sua taxa de aprendizagem deste modelo.

As perdas indicam que este modelo está a aprender de maneira eficaz ao longo do treino, com melhorias continuas na perda de localização, apresentadas na Figura 66 após os quinze mil passos acontece uma estabilização que se mantém até o fim do treino, convergindo no final do treino, comportamento que indica que o treino está a correr bem.

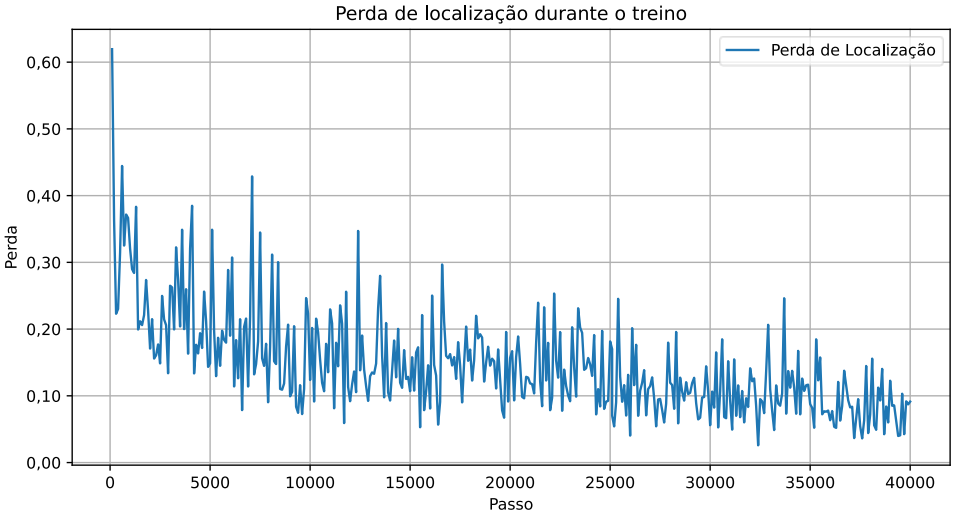


Figura 66 Perda de localização durante o treino – Modelo de deteção e classificação de laranjas - Duas classes

Para as perdas de classificação apresentadas na Figura 67, o modelo se mantém instável nos primeiros dez mil passos, após isso, a perda se estabiliza até o final, essa conversão, apesar de picos esporádicos, demostra que o modelo está sendo treinado de maneira adequada.

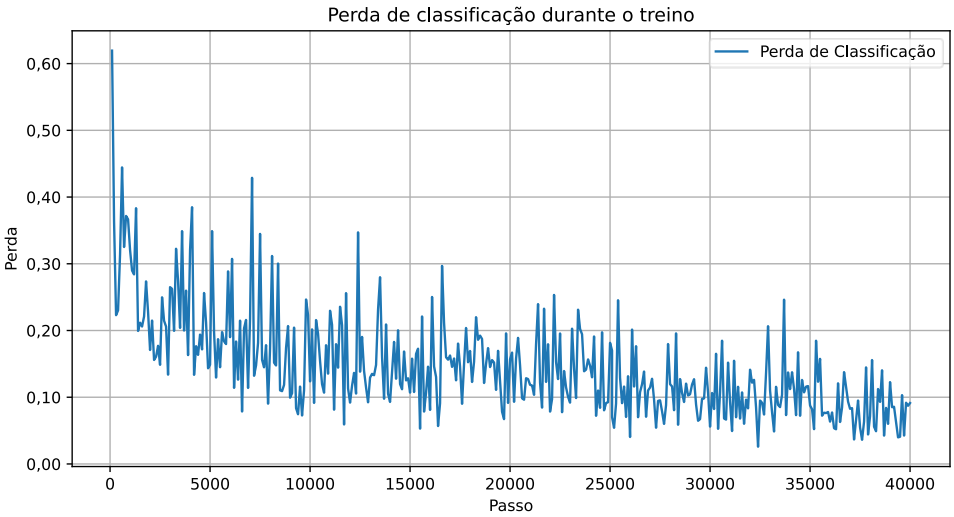


Figura 67 Perda de classificação durante o treino - Modelo de deteção e classificação de laranjas - Duas classes

A tendência decrescente apresentada pela Figura 68 indica que o modelo está se beneficiando da regularização, reduzindo o *overfitting* e melhorando sua generalização. A curva suave e contínua sugere uma diminuição estável da perda, sem grandes flutuações ou instabilidades.

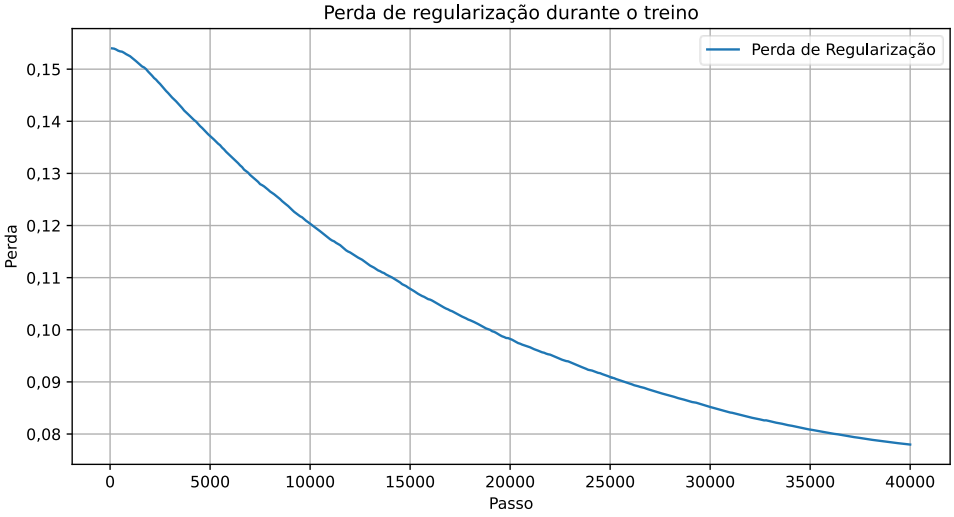


Figura 68 Perda de regularização durante o treino - Modelo de deteção e classificação de laranjas - Duas classes

As perdas totais apresentadas na Figura 69 inicialmente, a perda total é bastante alta, em torno de 1,20, mas ela diminui rapidamente nos primeiros 5.000 passos. A partir desse ponto, a perda continua a diminuir de forma mais gradual, com flutuações frequentes, refletindo variações no processo de aprendizado. Entre os 10.000 e 25.000 passos, a perda total oscila entre 0,20 e 0,60. Nos passos finais, a perda se estabiliza abaixo de 0,20, indicando que o modelo está convergindo para uma performance mais consistente.

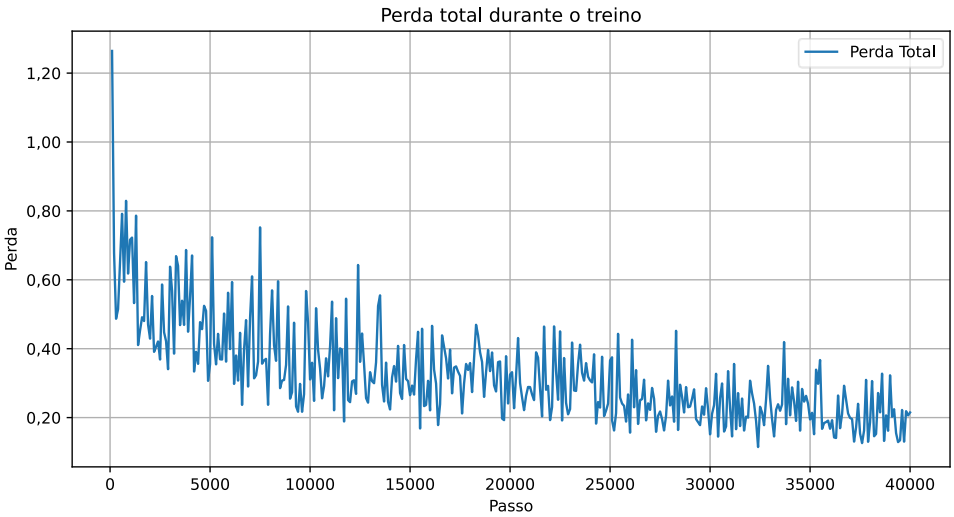


Figura 69 Perda de total durante o treino - Modelo de deteção e classificação de laranjas - Duas classes

Para a análise da taxa de aprendizagem, observa-se uma diminuição progressiva, permitindo ajustes finos nas etapas finais do treino (ver Figura 70).

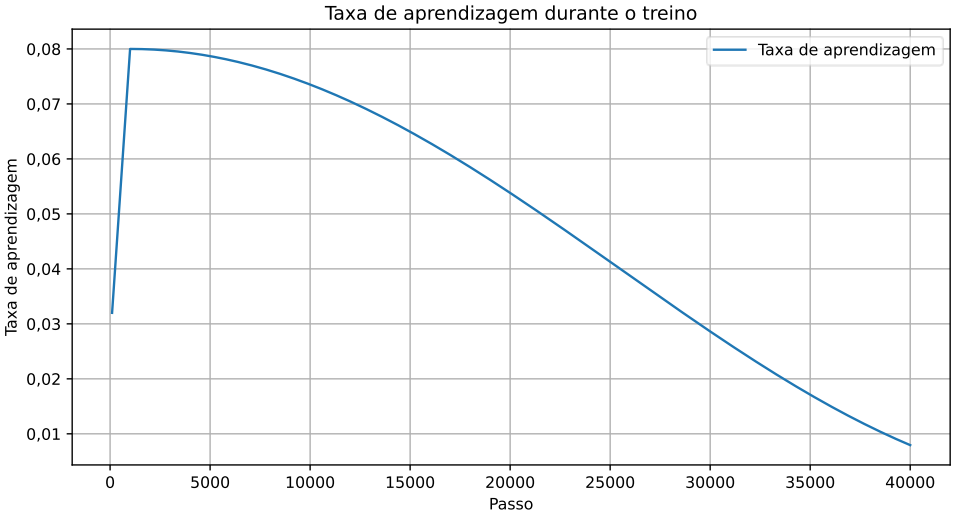


Figura 70 Taxa de aprendizagem - Modelo de deteção e classificação de laranjas - Duas classes

A Tabela 7 representa os resultados efetivos do modelo TensorFlow. A Tabela 8 apresenta os resultados obtidos após a conversão em TensorFlow Lite, esses resultados foram obtidos através dos métodos explicados na secção de avaliação do modelo.

Na análise numérica do treino do modelo de deteção e classificação de laranjas com 2 classes, baseado numa base de dados pública, foram utilizados gráficos e métricas que revelam a eficácia do modelo durante o treino. As métricas de precisão demonstram maior eficácia ao trabalhar com imagens grandes (71,6%) e desempenho inferior com imagens médias (33,5%) e pequenas (1,3%). A recuperação apresenta um

comportamento semelhante, com resultados superiores em imagens grandes (82,3%), seguidas por imagens médias (60%) e pequenas (1,3%).

O modelo apresenta uma precisão elevada (81%) com um limiar de IoU de 0,50, indicando eficácia na deteção de objetos com baixa sobreposição. Como esperado, a precisão diminui em limiares mais altos (67,6%). Em termos de recuperação, o modelo possui uma boa capacidade de detetar até 10 objetos (66,9%) e uma excelente capacidade de detetar até 100 objetos (74,3%).

Tabela 7 Resultados modelo TensorFlow - Modelo de deteção e classificação de laranjas - Duas classes

Métrica	IoU	Área	maxDets	Valor
Average Precision (AP)	[0,50:0,95]	all	100	0,616
	0,5	all	100	0,81
	0,75	all	100	0,676
	[0,50:0,95]	small	100	0,013
	[0,50:0,95]	medium	100	0,335
	[0,50:0,95]	large	100	0,716
Average Recall (AR)	[0,50:0,95]	all	1	0,369
	[0,50:0,95]	all	10	0,669
	[0,50:0,95]	all	100	0,743
	[0,50:0,95]	small	100	0,013
	[0,50:0,95]	medium	100	0,6
	[0,50:0,95]	large	100	0,823

Adicionalmente, os resultados após a conversão de TensorFlow Lite para diferentes limiares de IoU demonstram a variação na precisão média (mAP) entre laranjas frescas e podres. Com um limiar de IoU de 0,50, a precisão média alcançada foi de 69,85%, com uma AP de 57,41% para laranjas frescas e 82,29% para laranjas podres. À medida que o limiar de IoU aumenta, observa-se uma queda na mAP: 69,42% (IoU 0,55), 68,03% (IoU 0,60), 66,85% (IoU 0,65), e assim por diante, até chegar a apenas 7,54% em um IoU de 0,95. Deste modo obtendo um IoU [0,50:0,95] de 54,48%.

Tabela 8 Resultados modelo TensorFlow Lite - Modelo de deteção e classificação de laranjas - Duas classes

IoU	Fresh Orange AP (%)	Rotten Orange AP (%)	mAP (%)
0,5	57,41	82,29	69,85
0,55	56,62	82,22	69,42
0,6	54,56	81,49	68,03
0,65	52,54	81,16	66,85
0,7	48,4	80,76	64,58
0,75	43,04	78,68	60,86
0,8	33,49	77,53	55,51
0,85	21,26	73,12	47,19
0,9	9,6	60,27	34,93
0,95	0,42	14,66	7,54

Estes resultados correspondem à utilização deste modelo, na Figura 71 observa-se a sua aplicação em imagens que não foram utilizadas no seu treino.

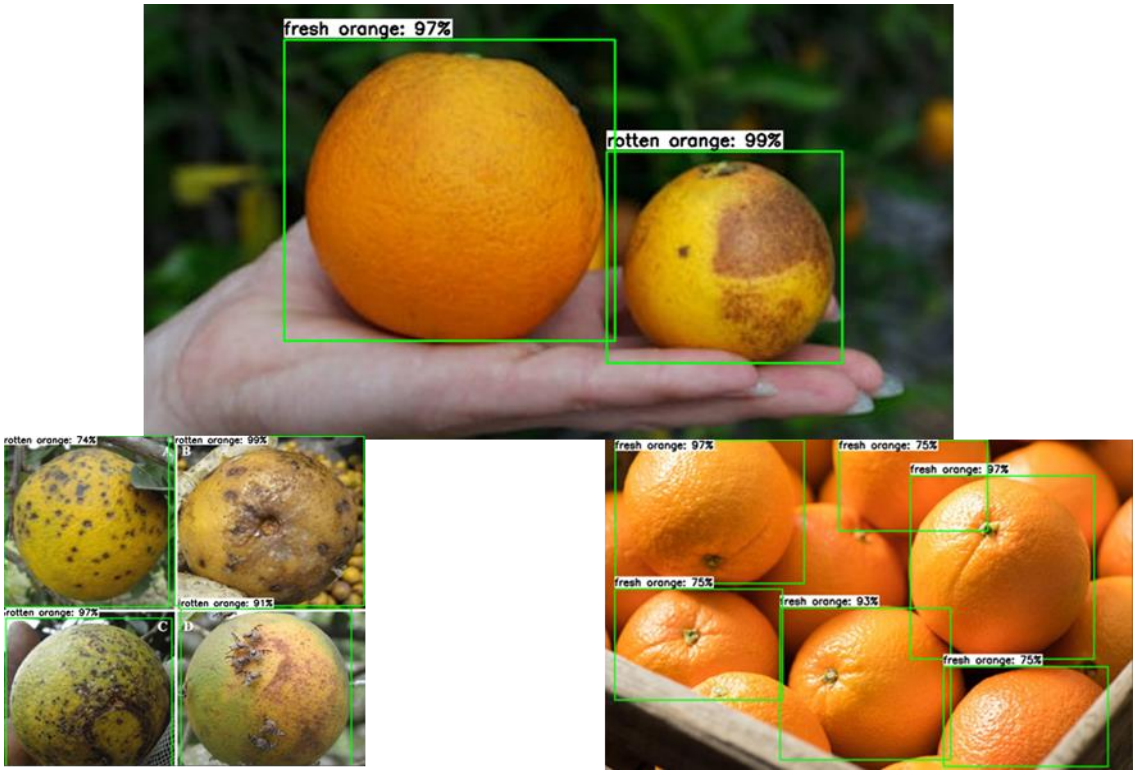


Figura 71 Modelo de deteção e classificação de laranjas com 2 classes, sendo aplicado em imagens fora da sua base de dados original

O desempenho deste modelo apresenta resultados positivos, mesmo aplicado em imagens com natureza diferente da sua base de dados de treino, demostrando assim efetividade em atividades de utilização reais.

4.1.1.2 Modelo de deteção e classificação de Laranja de 4 classes com base de dados científica

As Figuras 72 a 76 foram geradas com os dados obtidos pelo TensorBoard. As Figuras 72 a 75 mostram as perdas durante o treino do modelo TensorFlow, enquanto a Figura 76 exibe a taxa de aprendizagem deste mesmo modelo.

Inicialmente, a perda de classificação (vê Figura 72) é alta, alcançando um valor de cerca de 0,25, mas diminui rapidamente nos primeiros 2.000 passos. Após essa fase inicial, a perda se estabiliza significativamente abaixo de 0,05, com pequenas flutuações ao longo do restante do treino. Essa tendência indica que o modelo está aprendendo de forma eficiente, reduzindo a perda de classificação rapidamente e mantendo uma baixa taxa de erro durante o treino.

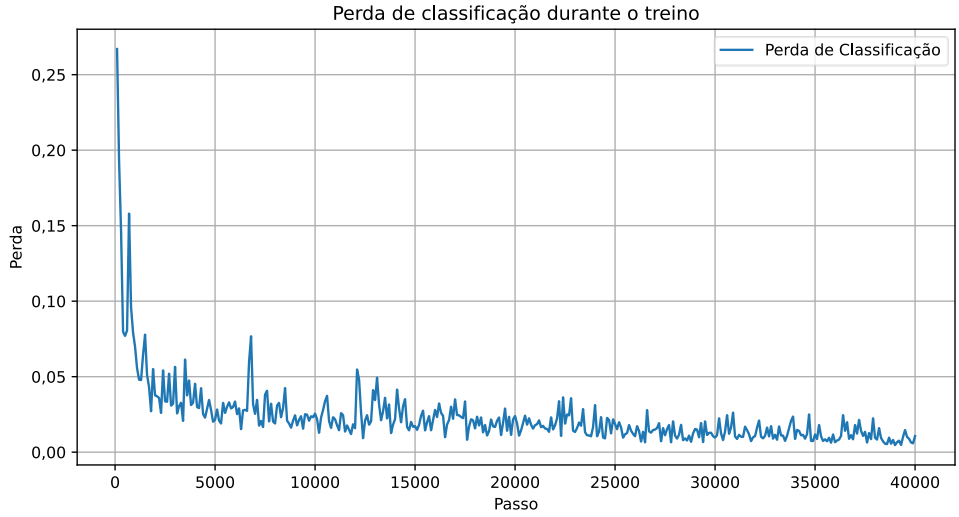


Figura 72 Perda de classificação durante o treino - Modelo de deteção e classificação de laranjas - Quatro classes

A perda de localização, apresentada na Figura 73, é relativamente alta, atingindo cerca de 0,07, mas ela diminui rapidamente nos primeiros 2.000 passos. Após essa fase inicial de rápida diminuição, a perda continua a reduzir de maneira mais gradual e estabiliza em valores próximos a 0,00, com pequenas flutuações. Essas flutuações são mais notáveis nos primeiros 10.000 passos, mas tornam-se menos frequentes e menos intensas à medida que o treino avança. Este padrão sugere que o modelo está melhorando sua capacidade de localização ao longo do tempo, tornando-se mais preciso e estável no processo.

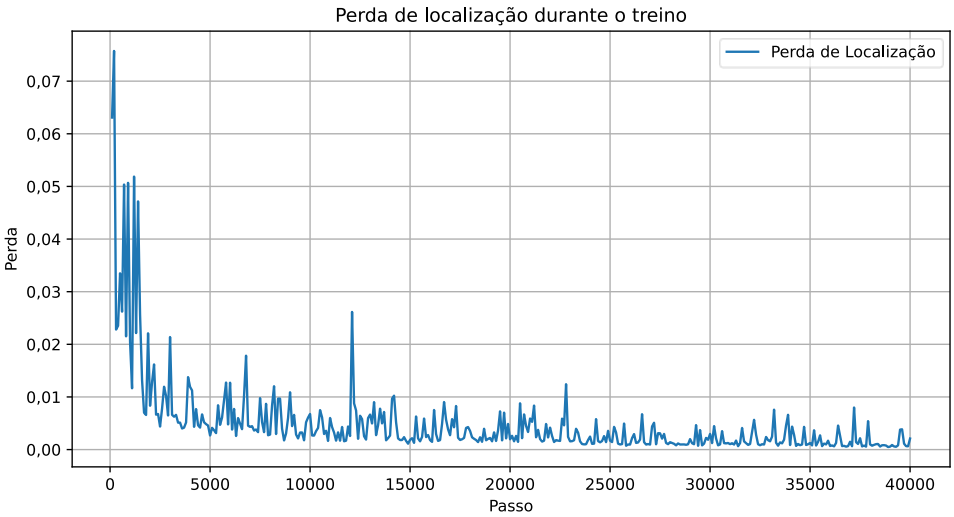


Figura 73 Perda de localização durante o treino - Modelo de deteção e classificação de laranjas - Quatro classes

A curva descendente suave e progressiva indica que o processo de regularização (vê Figura 74) está sendo efetivamente aplicado, ajudando a reduzir o *overfitting* e melhorar a generalização do modelo. A ausência de grandes flutuações sugere uma estabilização gradual e eficiente da perda de regularização, refletindo uma melhoria consistente na performance do modelo durante o treino.

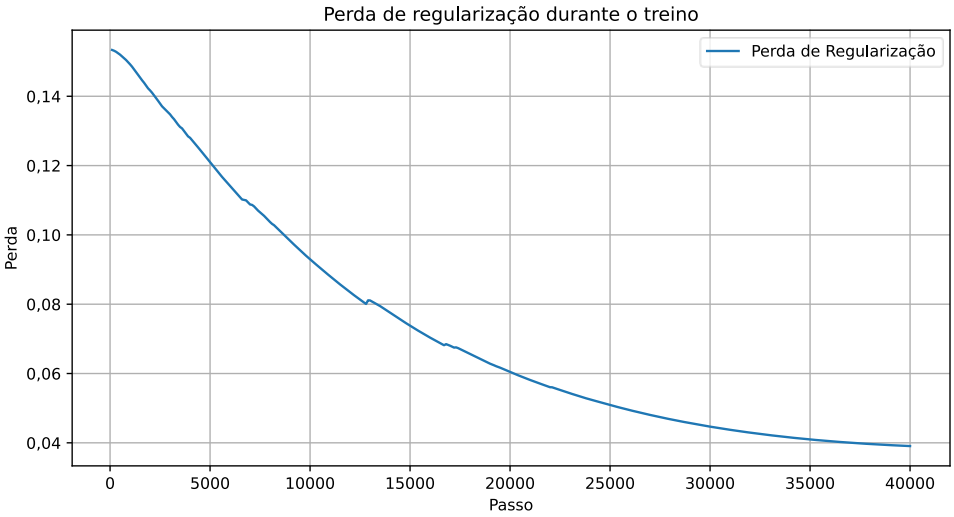


Figura 74 Perda de regularização durante o treino - Modelo de deteção e classificação de laranjas - Quatro classes

Após esse período inicial de declínio acentuado na perda total, ilustrada na Figura 75, a perda continua a diminuir de maneira mais gradual, com algumas flutuações, até se estabilizar abaixo de 0,10 nos passos finais. Essa tendência geral indica que o modelo está aprendendo de forma eficaz, melhorando seu desempenho ao longo do tempo de treino. A redução contínua e a estabilização da perda total sugerem que o modelo está convergindo para uma solução mais precisa e confiável.

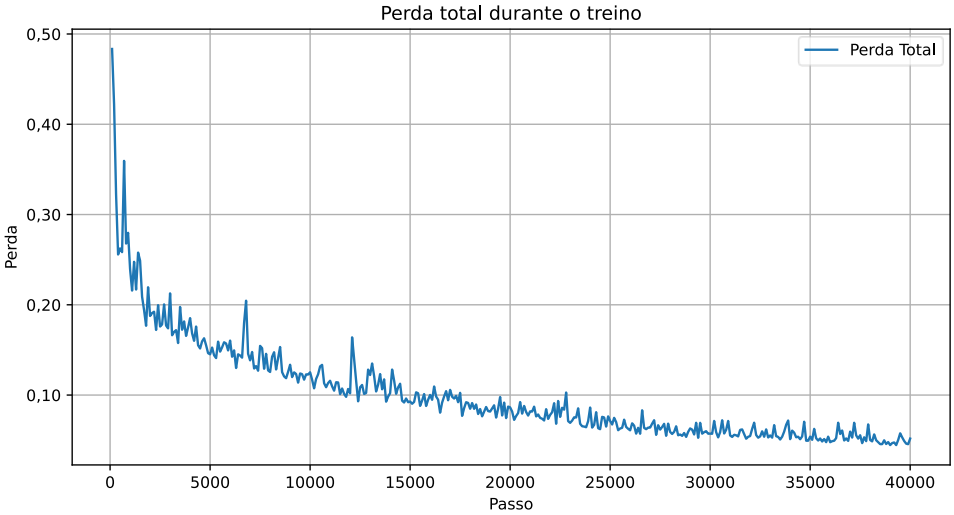


Figura 75 Perda total durante o treino - Modelo de deteção e classificação de laranjas - Quatro classes

Para a análise da taxa de aprendizagem, observa-se uma diminuição progressiva, permitindo ajustes precisos nas etapas finais do treino (ver Figura 76).

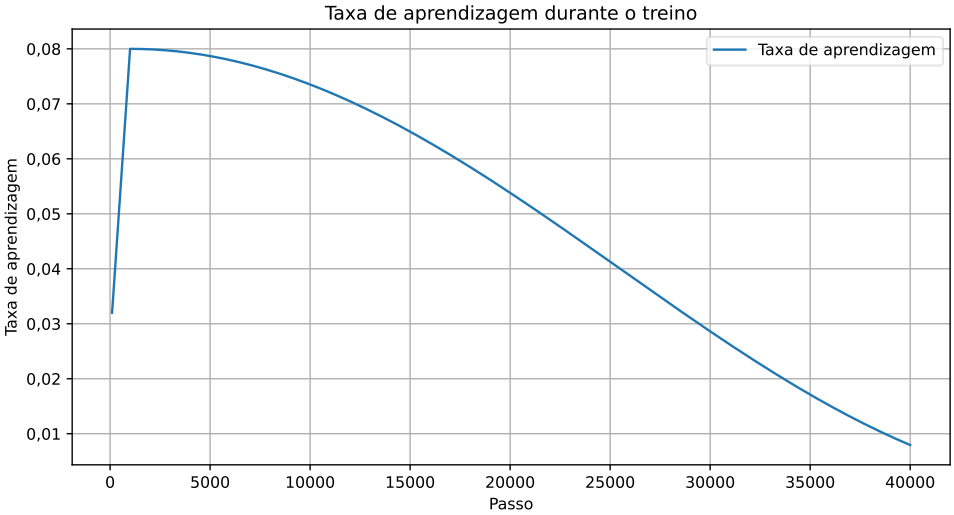


Figura 76 Taxa de aprendizagem durante o treino - Modelo de deteção e classificação de laranjas - Quatro classes

A Tabela 9 apresenta os resultados efetivos do modelo TensorFlow, enquanto a Tabela 10 mostra os resultados obtidos após a conversão para TensorFlow Lite. Esses resultados foram obtidos por meio dos métodos explicados na seção de avaliação do modelo.

Para a análise numérica do treino do modelo de deteção e classificação de laranjas com 4 classes, as métricas de precisão demonstram uma alta eficácia ao trabalhar com imagens grandes (98,0%) e desempenho inexistente para imagens médias e pequenas (indicadas como -1,000, sugerindo a ausência de deteções nessas categorias). A recuperação apresenta um comportamento semelhante, com resultados superiores em

imagens grandes (99,9%), enquanto imagens médias e pequenas também não apresentam deteções.

O modelo indica uma precisão média alta (98,0%) com um intervalo de IoU de [0,50:0,95], indicando eficácia na deteção de objetos com variação moderada de sobreposição. Com um IoU de 0,50, a precisão ligeiramente aumenta para 98,1%, sugerindo excelente desempenho em deteções com baixa sobreposição. A precisão mantém-se estável em valores de IoU mais elevados, como 0,75, com um valor de 98,1%. Em termos de recuperação, indica-se uma alta capacidade de detetar até 1, 10 e 100 objetos, todos com um valor de 99,9%. Isto indica uma excelente habilidade do modelo em recuperar a maioria dos objetos presentes nas imagens analisadas.

Tabela 9 Resultados modelo TensorFlow - Modelo de deteção e classificação de laranjas - Quatro classes

Métrica	IoU	Área	maxDets	Valor
Average Precision (AP)	[0,50:0,95]	all	100	0,98
	0,5	all	100	0,981
	0,75	all	100	0,981
	[0,50:0,95]	small	100	-1
	[0,50:0,95]	medium	100	-1
	[0,50:0,95]	large	100	0,98
Average Recall (AR)	[0,50:0,95]	all	1	0,999
	[0,50:0,95]	all	10	0,999
	[0,50:0,95]	all	100	0,999
	[0,50:0,95]	small	100	-1
	[0,50:0,95]	medium	100	-1
	[0,50:0,95]	large	100	0,999

Após a conversão para TensorFlow Lite, os resultados demonstram que o modelo é incapaz de reconhecer duas das quatro classes existentes. Esse desempenho contrasta com os resultados pré-conversão, que indicavam um modelo praticamente perfeito. Mesmo considerando as perdas naturais de precisão durante a conversão, a incapacidade de reconhecer metade das classes sugere que os resultados anteriores podem ter proporcionado uma falsa sensação de qualidade do modelo. Isso ocorre porque a base de dados utilizada apresenta pouquíssimas diferenças entre os elementos, mesmo entre classes diferentes, e possui uma quantidade limitada de dados, o que reduz a eficácia do modelo em detetar objetos que se desviem minimamente de seus parâmetros. Outro fator que demostra como este resultado deve ser descartado é o valor estático do mAP independente do IoU, comportamento completamente diferente dos outros modelos, o que faz um mAP para a faixa IoU [0,5:0,95] de 46,33%.

Tabela 10 Resultados modelo TensorFlow Lite - Modelo de deteção e classificação de laranjas - Quatro classes

IoU	Blackspot AP	Fresh AP	Grenning AP	Canker AP	mAP
0,5	95,99	0	0	89,49	46,37
0,55	95,99	0	0	89,49	46,37
0,6	95,99	0	0	89,49	46,37
0,65	95,99	0	0	89,49	46,37
0,7	95,99	0	0	89,49	46,37
0,75	95,99	0	0	89,49	46,37
0,8	95,99	0	0	89,49	46,37
0,85	95,99	0	0	89,49	46,37
0,9	95,99	0	0	89,49	46,37
0,95	95,99	0	0	87,72	45,9275

Como se vê na Figura 77, contrariando os resultados matemáticos do primeiro método de avaliação, este modelo está com resultados completamente distorcidos devido à origem da sua base de dados, tornando-o num modelo não fiável para o tipo de aplicação que se pretende neste projeto.

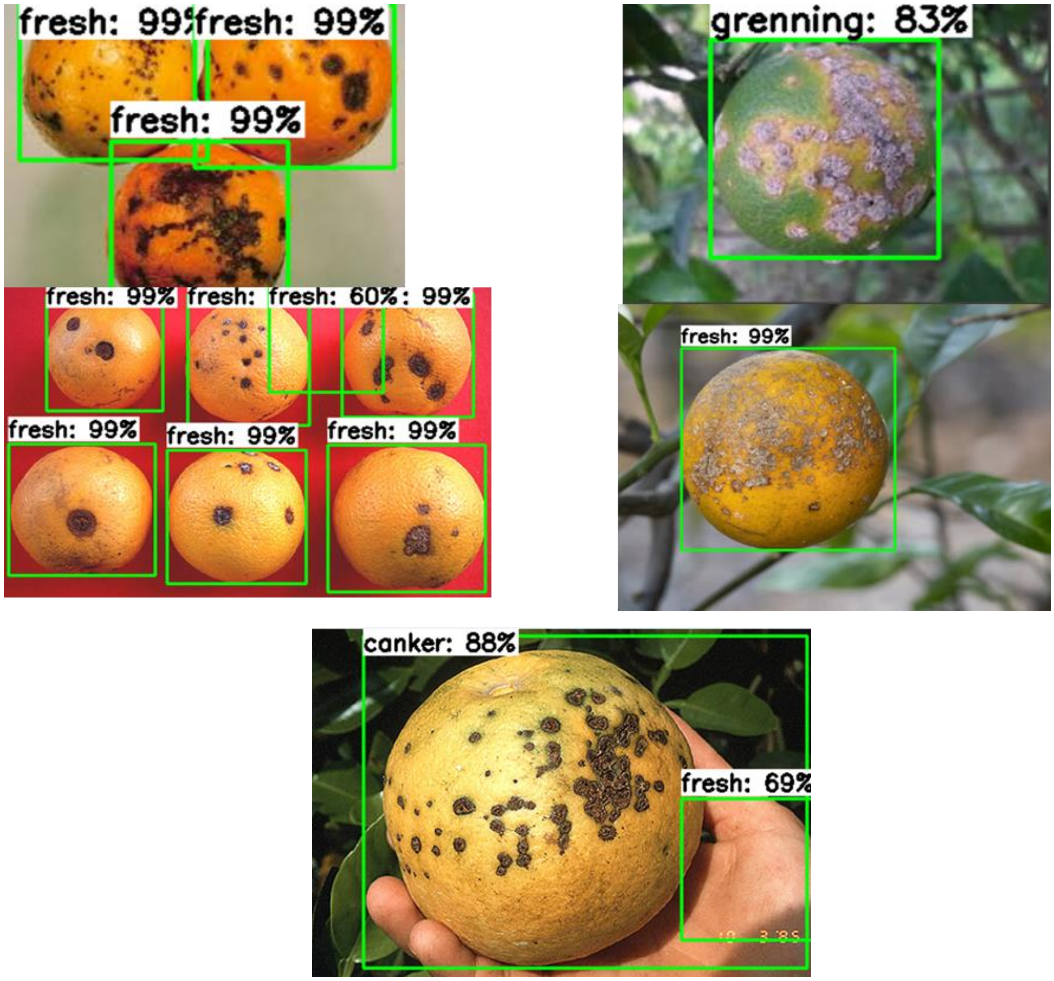


Figura 77 Modelo de deteção e classificação de laranjas com 4 classes sendo aplicado em imagens fora da sua base de dados original

O resultado em figuras que fogem da natureza da base de dados não teve bons resultados, demostrando falta de precisão e confundindo classes, esse modelo não seria útil em uma atuação na vida real.

4.1.2 Modelo de deteção e classificação de Tomate

O modelo para deteção de tomate foi dividido em dois principais grupos, um com duas classes e outro com quatro classes, utilizando uma base de dados pública retirada do Kaggle e Roboflow Universe. Os formatos gerais destes modelos foram explicados na secção 3.11.1.

4.1.2.1 Modelo de deteção e classificação de tomate com quatro classes

As Figuras 78 a 81 foram criadas com dados obtidos pelo TensorBoard. As Figuras 78 a 80 ilustram as perdas durante o treino do modelo TensorFlow.

Inicialmente, a perda de classificação (vê Figura 78) é relativamente alta, em torno de 0,35, mas diminui de forma acentuada nos primeiros 5.000 passos. Após essa fase inicial, a perda de classificação continua a diminuir, mas de forma mais gradual, com flutuações consideráveis ao longo do treino. A partir de aproximadamente 10.000 passos, a perda se estabiliza em torno de valores mais baixos, abaixo de 0,10, embora ainda apresente picos esporádicos. Esta tendência indica que o modelo está aprendendo e melhorando sua capacidade de classificação, mas ainda apresenta variações que podem ser resultado de complexidades nos dados de treino ou no processo de ajuste do modelo.

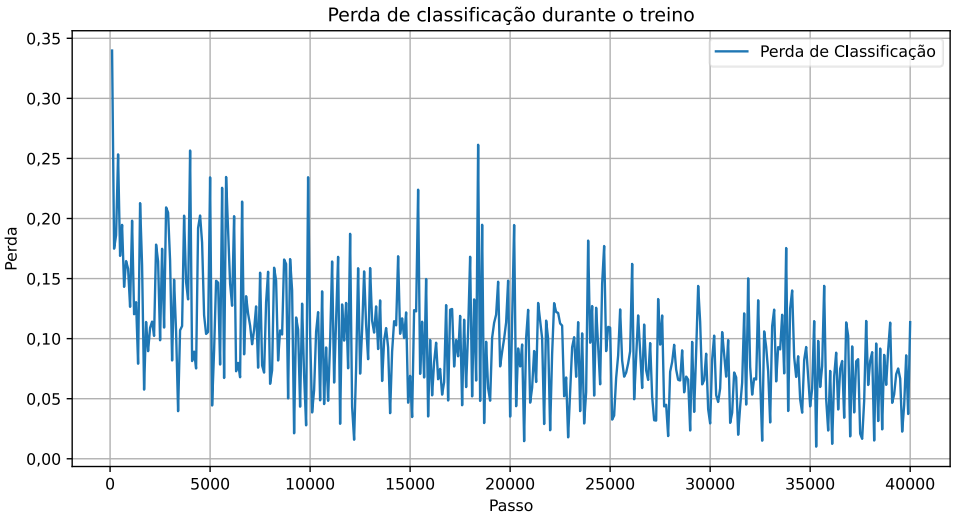


Figura 78 Perda de classificação durante o treino – Modelo de deteção e classificação de tomate - Quatro classes

A perda de localização (vê Figura 79) é alta, em torno de 0,40, mas diminui rapidamente nos primeiros 5.000 passos. Após essa fase inicial, a perda continua a diminuir de forma mais gradual, com flutuações significativas ao longo do tempo. A perda se estabiliza em torno de valores mais baixos, variando principalmente entre 0,05 e 0,20 após os primeiros 10.000 passos. No entanto, a presença de picos esporádicos ao longo de todo o período de treino sugere que há variabilidade nos dados ou no processo de ajuste do modelo.

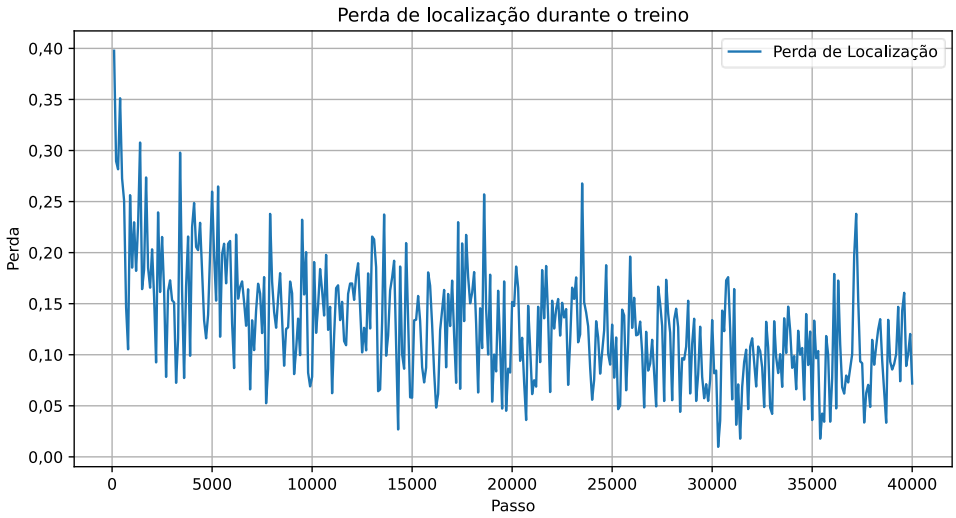


Figura 79 Perda de localização durante o treino - Modelo de deteção e classificação de tomate - Quatro classes

A perda total (vê Figura 80) é bastante alta, em torno de 0,90, mas diminui rapidamente nos primeiros 5.000 passos. Após esse período inicial, a perda continua a diminuir de forma mais gradual, com várias flutuações ao longo do tempo. Entre os 10.000 e 20.000 passos, a perda total varia significativamente entre 0,30 e 0,50, mas continua a tendência geral de diminuição. Nos passos finais, a perda estabiliza em torno de 0,20, com menos variabilidade. Esta tendência geral de redução indica que o modelo está aprendendo e ajustando-se melhor aos dados ao longo do tempo de treino, embora ainda haja alguma instabilidade, possivelmente devido à complexidade dos dados ou ajustes do modelo.

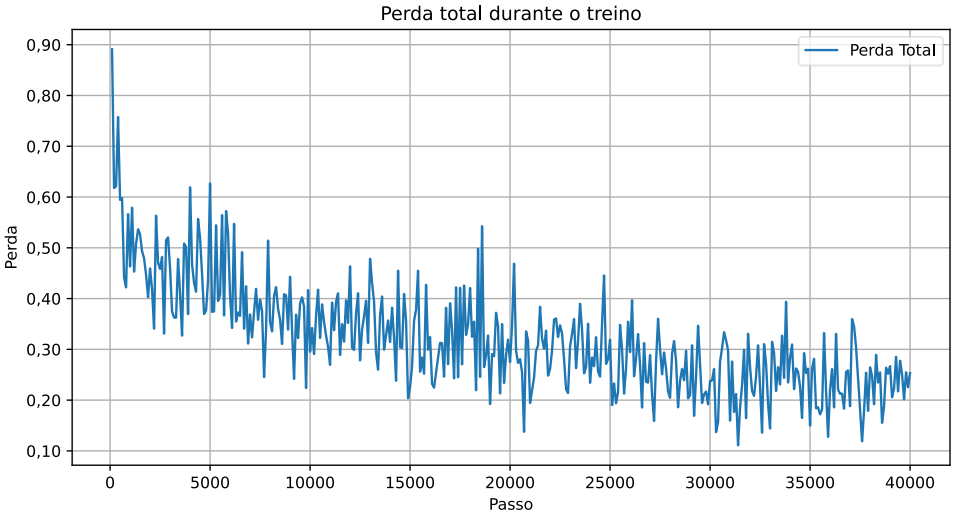


Figura 80 Perda total durante o treino - Modelo de deteção e classificação de tomate - Quatro classes

A perda de regularização (vê Figura 81) começa em torno de 0,15 e diminui de maneira constante e suave ao longo do tempo de treino, atingindo aproximadamente 0,07 no final. A curva descendente contínua e estável indica que o processo de regularização está sendo aplicado eficazmente, ajudando a reduzir o overfitting e melhorando a generalização do modelo.

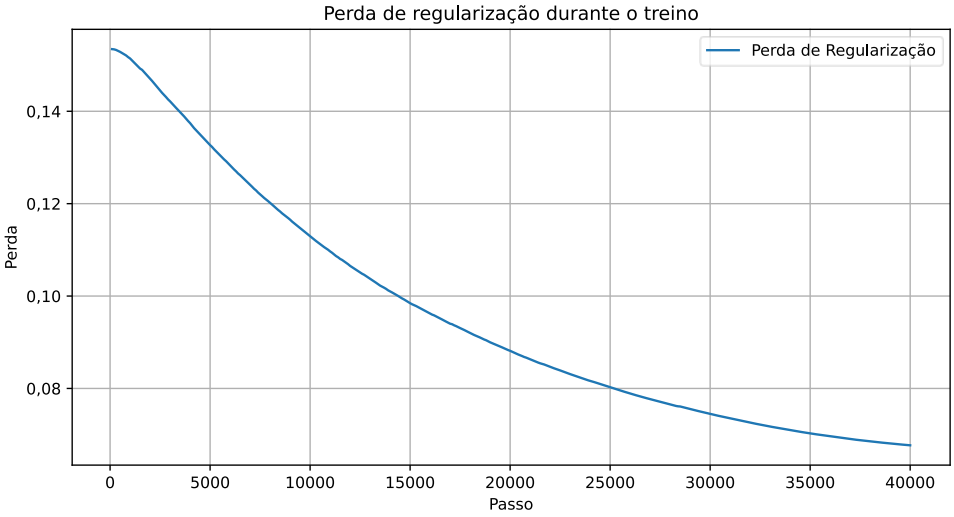


Figura 81 Perda de regularização durante o treino - Modelo de deteção e classificação de tomate - Quatro classes

A Figura 82 mostra a taxa de aprendizagem deste modelo. A taxa de aprendizagem, novamente, observa-se uma diminuição progressiva, permitindo ajustes finos nas etapas finais do treino.

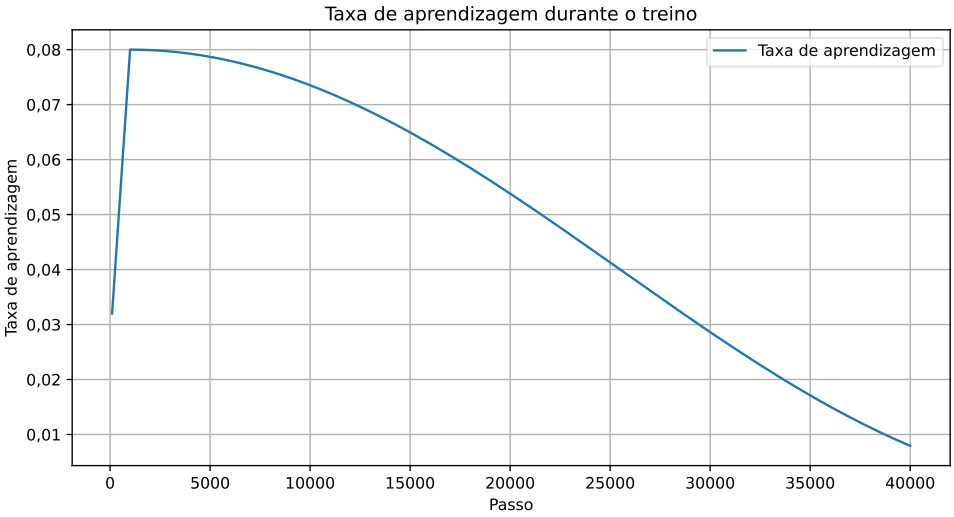


Figura 82 Taxa de aprendizagem durante o treino - Modelo de deteção e classificação de tomate - Quatro classes

A Tabela 11 representa os resultados efetivos do modelo TensorFlow, a Tabela 12 exibe os resultados após a conversão para TensorFlow Lite. Estes resultados foram obtidos utilizando os métodos descritos na seção de avaliação do modelo.

Para a análise numérica do treino do modelo de deteção e classificação de tomate com 4 classes, as métricas de precisão demonstram maior eficácia ao trabalhar com imagens grandes (77,4%) e desempenho inferior para imagens médias (41,6%) e nenhuma eficácia para imagens pequenas (0,0%). A recuperação apresenta um comportamento semelhante, com resultados superiores em imagens grandes (82,5%), seguidas por imagens médias (59,9%) e nenhuma recuperação para imagens pequenas (0,0%).

O modelo apresenta uma precisão alta (91,1%) com um limiar de IoU de 0,50, indicando uma excelente eficácia na deteção de objetos com baixa sobreposição. Como esperado, a precisão diminui em limiares mais elevados (83,6%). Em termos de recuperação, o modelo possui uma capacidade moderada de detetar até 1 objeto (29,0%) e uma boa capacidade de detetar até 10 objetos (72,9%) e até 100 objetos (80,1%).

Tabela 11 Resultados modelo TensorFlow - Modelo de deteção e classificação de tomate - Quatro classes

Métrica	IoU	Área	maxDets	Valor
Average Precision (AP)	[0,50:0,95]	all	100	0,738
	0,5	all	100	0,911
	0,75	all	100	0,836
	[0,50:0,95]	small	100	0
	[0,50:0,95]	medium	100	0,416
	[0,50:0,95]	large	100	0,774
Average Recall (AR)	[0,50:0,95]	all	1	0,29
	[0,50:0,95]	all	10	0,729
	[0,50:0,95]	all	100	0,801
	[0,50:0,95]	small	100	0
	[0,50:0,95]	medium	100	0,599
	[0,50:0,95]	large	100	0,825

A conversão do TensorFlow Lite para diferentes limiares de IoU demonstra a variação na precisão média (mAP) entre tomates em diferentes estados de maturação. Com um limiar de IoU de 0,50, a precisão média alcançada foi de 78,69%, com uma AP de 65,49% para tomates meio maduros, 90,72% para tomates muito maduros, 68,82% para tomates maduros e 89,74% para tomates podres. À medida que o limiar de IoU aumenta, observa-se uma queda na mAP: 78,59% (IoU 0,55), 77,97% (IoU 0,60), 77,86% (IoU 0,65), 77,15% (IoU 0,70), e assim por diante, até chegar a apenas 16,93% em um IoU de 0,95. Deste modo, este modelo apresentou um mAP para a faixa de IoU [0.5:0.95] de 67,54%. Este resultado apresenta-se válido em aplicações reais, como se observa na Figura 83.

Tabela 12 Resultados modelo TensorFlow Lite - Modelo de deteção e classificação de tomate - Quatro classes

IoU	Tomato Half Ripe AP	Tomato Overripe AP	Tomato Ripe AP	Tomato Rotten AP	mAP
0,5	65,49	90,72	68,82	89,74	78,69
0,55	65,34	90,72	68,55	89,74	78,59
0,6	65,2	89,7	67,35	89,62	77,97
0,65	65,2	89,7	67,03	89,49	77,86
0,7	64,63	87,95	66,67	89,34	77,15
0,75	63,06	87,01	65,93	88,67	76,17
0,8	60,99	87,01	62,72	86,82	74,38
0,85	52,41	82,59	56,11	82,24	68,34
0,9	32,63	65,25	34,17	65,03	49,27
0,95	4,27	25,6	13,54	24,32	16,93

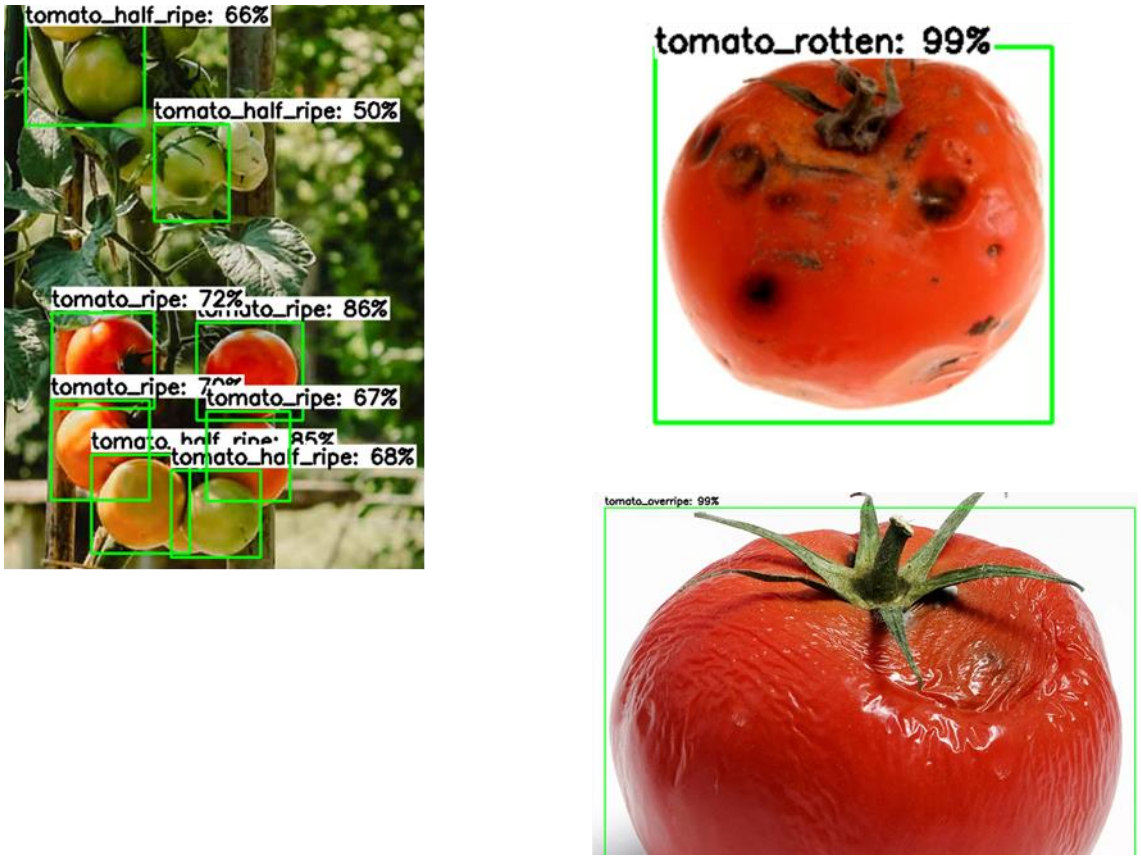


Figura 83 Modelo de deteção e classificação de tomates com 4 classes aplicado em imagens fora da sua base de dados original

O modelo trabalhou bem em imagens fora da natureza da base de dados que serviu para seu treino, deste modo, se apresentando viável para utilizações na vida real.

4.1.2.2 Modelo de deteção e classificação de Tomate com duas classes

As Figuras 84 a 88 foram criadas a partir dos dados obtidos pelo TensorBoard. As Figuras 88 a 87 detalham as perdas durante o treino do modelo TensorFlow, enquanto a Figura 88 destaca a taxa de aprendizagem do modelo.

Observa-se uma tendência geral de diminuição na perda por classificação ilustrada na Figura 84, indicando que o modelo está aprendendo e melhorando. No entanto, há oscilações e picos significativos, sugerindo dificuldades momentâneas no processo de aprendizagem. A partir de certo ponto, a perda se estabiliza, indicando que o modelo alcançou um platô de desempenho. Esse comportamento é típico e pode guiar ajustes para otimizar ainda mais o treino.

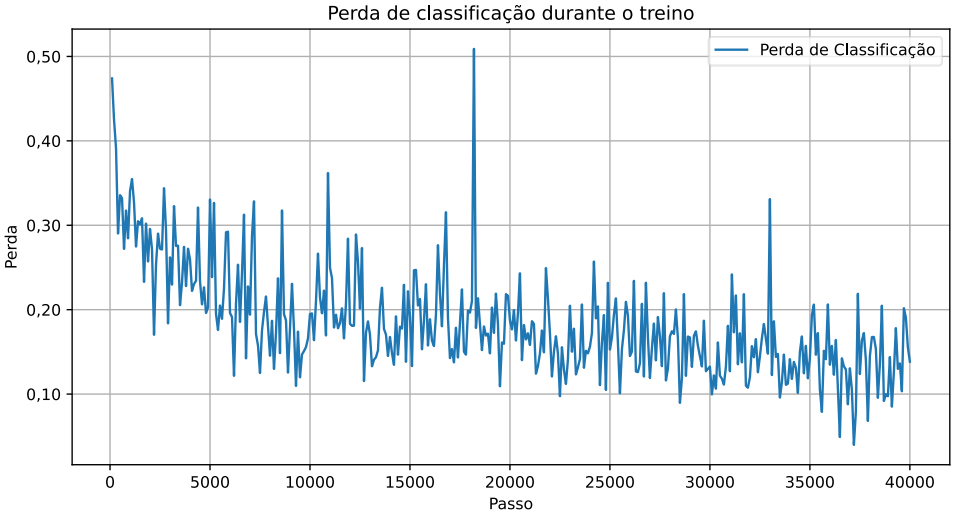


Figura 84 Perda de classificação durante o treino - Modelo de deteção e classificação de tomate - Duas classes

Observa-se uma diminuição geral na perda de localização, indicando que o modelo está se tornando mais preciso na tarefa de localização. No entanto, há flutuações e alguns picos ao longo do treino, sugerindo momentos de dificuldade. A perda se estabiliza em um valor baixo, indicando que o modelo atingiu um platô de desempenho. Esse padrão é comum e pode ajudar a identificar pontos para ajustes no processo de treino.

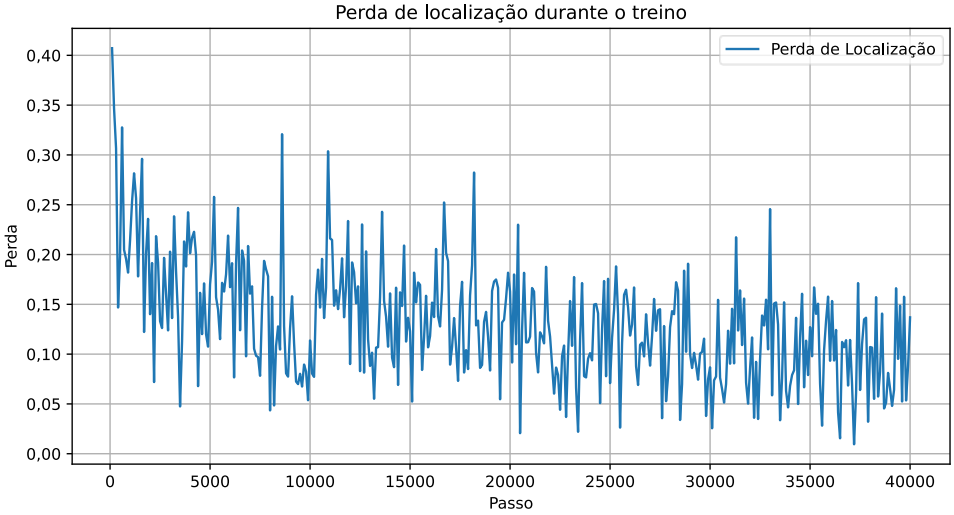


Figura 85 Perda de localização durante o treino - Modelo de deteção e classificação de tomate - Duas classes

Nota-se uma tendência geral de queda na perda total (vê Figura 86), indicando que o modelo está aprendendo e melhorando sua precisão. Contudo, existem flutuações e picos notáveis, refletindo variações na dificuldade dos dados de treino. Apesar das oscilações, a perda se estabiliza em valores mais baixos, sugerindo que o modelo alcançou um platô de desempenho. Esse comportamento é esperado e útil para orientar possíveis ajustes no processo de treino para otimizar os resultados.

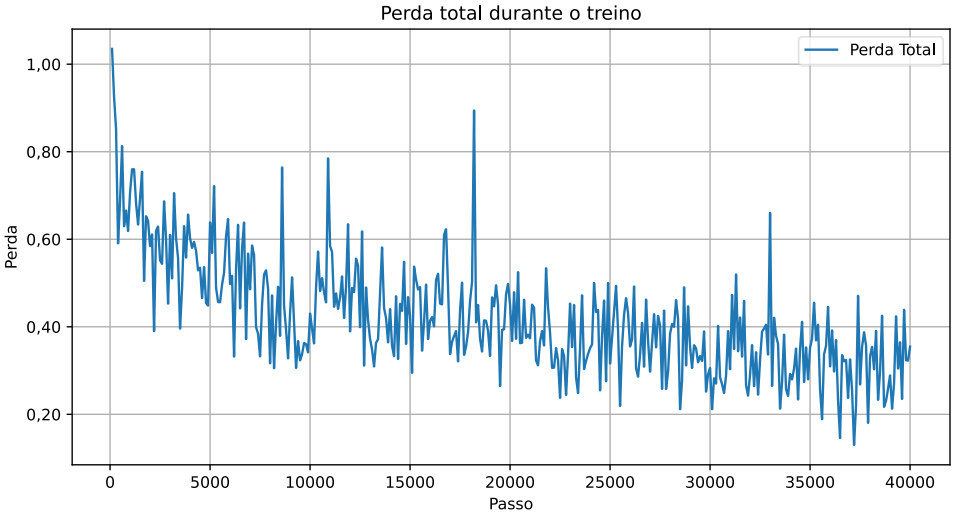


Figura 86 Perda total durante o treino - Modelo de deteção e classificação de tomate - Duas classes

O comportamento indicado pela Figura 87, demostra que a regularização está cumprindo seu papel de reduzir a complexidade do modelo, prevenindo o *overfitting* sem grandes oscilações. A tendência descendente consistente sugere uma melhoria gradual na generalização do modelo, tornando-o mais robusto e menos suscetível a se ajustar excessivamente aos dados de treino.

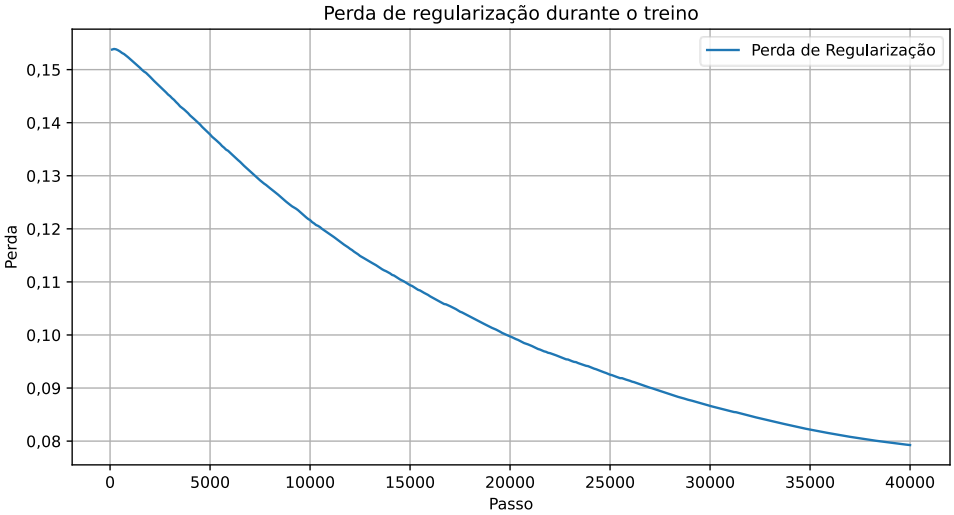


Figura 87 Perda de regularização durante o treino - Modelo de deteção e classificação de tomate - Duas classes

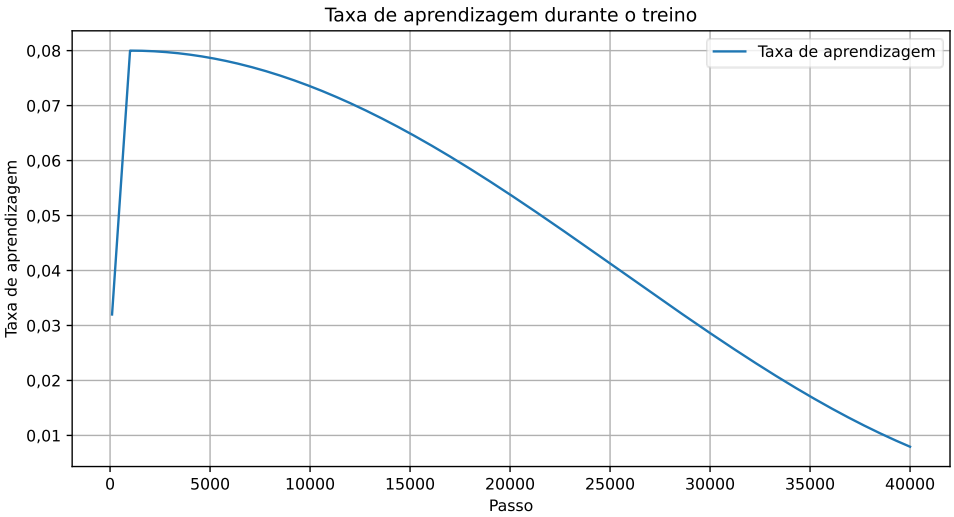


Figura 88 Taxa de aprendizagem durante o treino - Modelo de deteção e classificação de tomate - Duas classes

A Tabela 13 representa os resultados efetivos do modelo TensorFlow, a Tabela 14 apresenta os resultados obtidos após a conversão em TensorFlow Lite. Estes resultados foram obtidos através dos métodos explicados na secção de avaliação do modelo.

Para a análise numérica do treino do modelo de deteção e classificação de tomate com 2 classes, as métricas de precisão demonstram maior eficácia ao trabalhar com imagens grandes (83,6%) e desempenho inferior para imagens médias (62,7%) e pequenas (6,5%). A recuperação apresenta um comportamento semelhante, com resultados superiores em imagens grandes (79,2%), seguidas por imagens médias (73,4%) e pequenas (22,2%).

O modelo apresenta uma precisão alta (91,5%) com um limiar de IoU de 0,50, indicando eficácia na deteção de objetos com baixa sobreposição. Como esperado, a precisão diminui em limiares mais elevados (82,5%). Em termos de recuperação, o modelo possui uma boa capacidade de detetar até 10 objetos (67,5%) e uma excelente capacidade de detetar até 100 objetos (79,2%).

Tabela 13 Resultados modelo TensorFlow - Modelo de deteção e classificação de tomate - Duas classes

Métrica	IoU	Área	maxDets	Valor
Average Precision (AP)	[0,50:0,95]	all	100	0,719
	0,5	all	100	0,915
	0,75	all	100	0,825
	[0,50:0,95]	small	100	0,065
	[0,50:0,95]	medium	100	0,627
	[0,50:0,95]	large	100	0,836
Average Recall (AR)	[0,50:0,95]	all	1	0,253
	[0,50:0,95]	all	10	0,675
	[0,50:0,95]	all	100	0,792
	[0,50:0,95]	small	100	0,222
	[0,50:0,95]	medium	100	0,734
	[0,50:0,95]	large	100	0,878

Com a conversão ao TensorFlow Lite demonstra a variação na precisão média (mAP) esperada entre as duas classes de tomates. Com um limiar de IoU de 0,50, a precisão média alcançada foi de 81,43%, com uma AP de 72,32% para tomate saudável e 90,54% para tomate podre. À medida que o limiar de IoU aumenta, observa-se uma queda na mAP: 80,055% (IoU 0,55), 79,685% (IoU 0,60), 79,14% (IoU 0,65), 77,82% (IoU 0,70), e assim por diante, até chegar a apenas 3,82% em um IoU de 0,95. Deste modo, o valor do mAP para IoU [0,5:0,95] de 64,66%. Este resultado se apresenta valido em aplicações reais, como se observa na Figura 89.

Tabela 14 Resultados modelo TensorFlow Lite - Modelo de deteção e classificação de tomate - Duas classes

IoU	tomato_healthy AP (%)	tomato_rotten AP (%)	mAP (%)
0,5	72,32	90,54	81,43
0,55	70,02	90,09	80,055
0,6	69,84	89,53	79,685
0,65	69,23	89,05	79,14
0,7	68,21	87,43	77,82
0,75	67,01	84,61	75,81
0,8	60,12	80,76	70,44
0,85	55,84	72,93	64,385
0,9	21,04	46,91	33,975
0,95	1,32	6,32	3,82

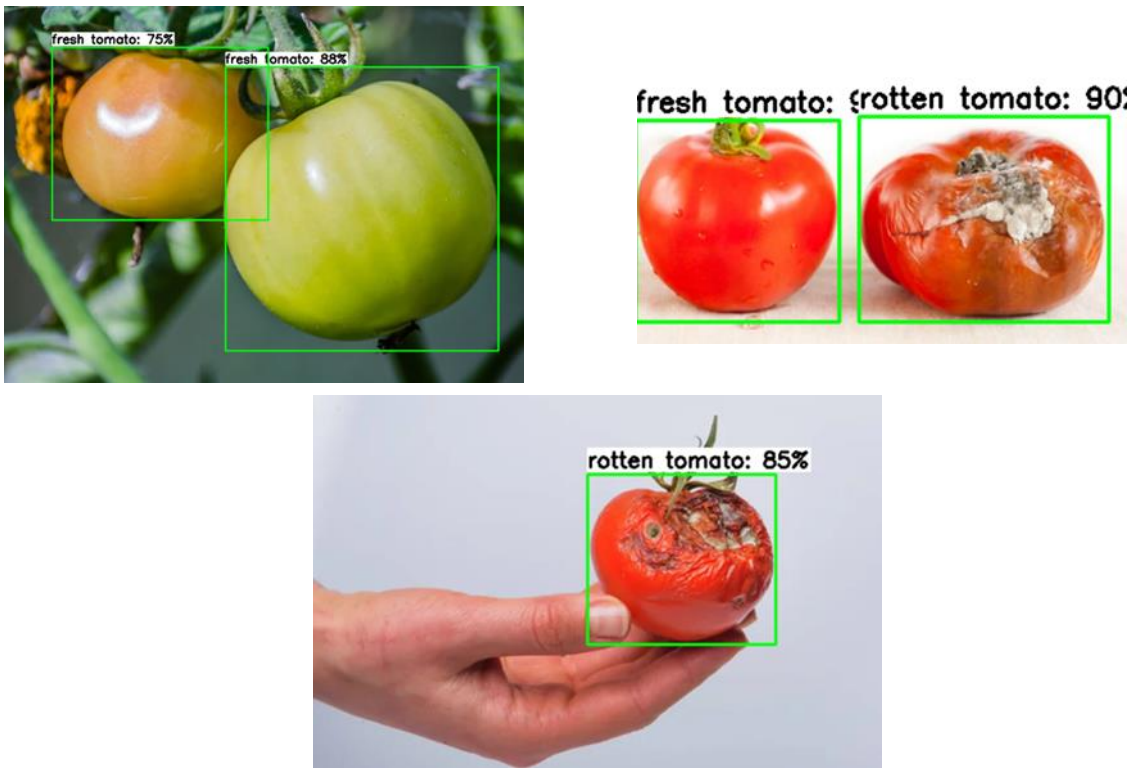


Figura 89 Modelo de deteção e classificação de tomate com 2 classes aplicado em imagens fora da sua base de dados original

Os resultados relativos ao tamanho estão de acordo com a natureza da base de dados utilizada, onde a maioria das amostras possuía anotações de imagens grandes.

Para os testes do sistema, todas as amostras possuem áreas maior que 96^2 pixéis, devido à distância da câmara à área de trabalho, a resolução da câmara e a natureza em si do tipo de fruto analisado. Então, os resultados a serem analisados para este sistema são sempre as das imagens grandes, como demostrado na Figura 90.

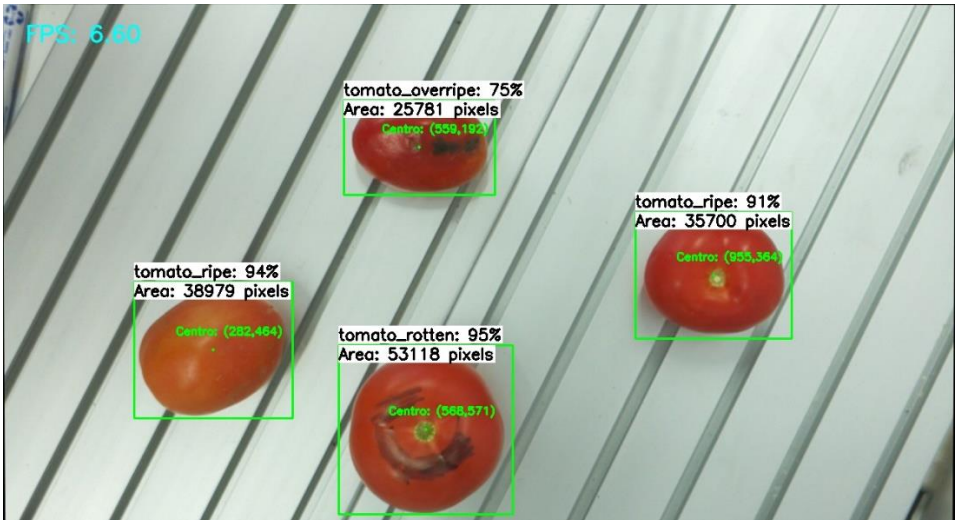


Figura 90 Demostração do tamanho da área dos objetos em pixéis.

Este modelo apresentou bons resultados para imagens fora da natureza de sua base de dados, sendo assim, um modelo valido para utilização real.

4.2 Teste do sistema

A Figura 91 apresenta o resultado das 640 tentativas para apanhar o fruto. Para este teste foi utilizado o modelo de deteção e classificação de tomate com duas classes, foi considerado sucesso quando a garra foi capaz de apanhar o fruto e levá-lo até à área determinada (pontos verdes na Figura 91). Foi considerada uma área de cinemática perigosa quando o braço ficou relativamente próximo de uma colisão (pontos amarelos na Figura 91) e foi considerado uma colisão sempre que o sistema anticolisão do UR3e foi ativo ou quando a mensagem de cinematicamente impossível foi acionada (pontos vermelhos na Figura 91).

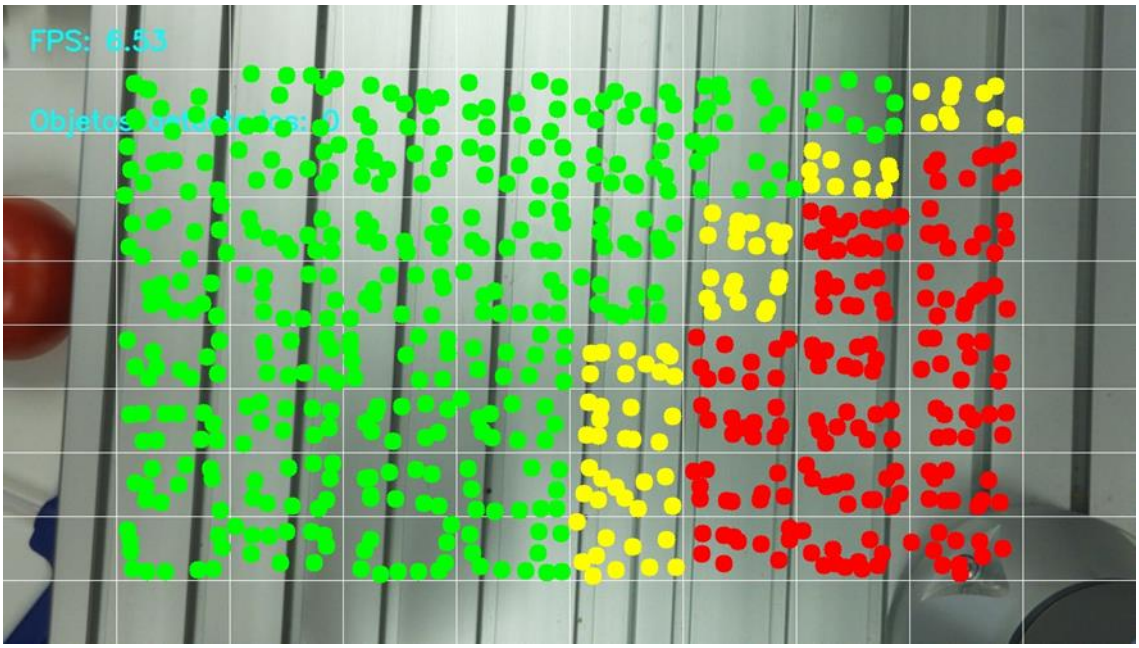


Figura 91 Resultados do sistema

Para os testes do sistema, das 640 tentativas de manipulação, 390 obtiveram sucesso de forma segura, 80 obtiveram sucesso, mas operando demasiado próximo do limite da área de trabalho, e as 170 restantes fizeram o sistema de colisão do UR3e ativar. Deste modo, a fim de criar uma área de trabalho fiável de 65500 mm^2 ao sistema, será utilizado a zona de trabalho destacada na Figura 92.

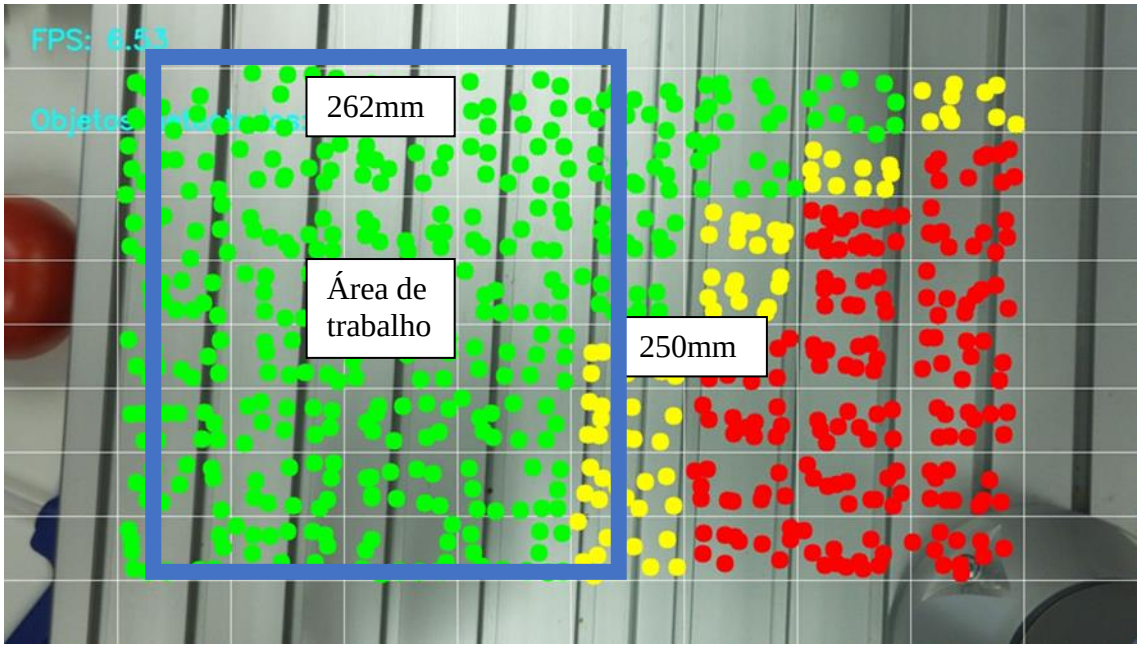


Figura 92 Configuração da área de trabalho proposta

4.3 Nota conclusiva

Neste capítulo, foi abordado o processo de treino dos modelos de deteção e classificação de laranjas (4 classes e 2 classes) e de tomates (4 classes e 2 classes), detalhando os resultados obtidos, sendo eles as perdas de classificação, localização, regularização e total referentes ao treino, os resultados de precisão e recuperação com base no IoU e tamanho da imagem ao fim do treino como modelo TensorFlow e por fim os resultados em base ao limite IoU e classe após a conversão para modelo TensorFlow Lite. O desempenho individual de cada modelo é examinado, destacando a perda natural de precisão no processo de conversão para um modelo TensorFlow Lite. Além disso, é debatida a eficácia dos modelos quando testados em imagens que não pertencem ao seu conjunto de dados original. Três dos quatro modelos treinados foram bem-sucedidos, com especial atenção para a avaliação enganosa do modelo de deteção de laranjas com 4 classes, que matematicamente aparentava ser um bom modelo, mas teve péssimos resultados em imagens fora da natureza da sua base de dados. A escolha de uma área retangular e a exclusão da restante da zona que apresentou resultados positivos, baseia-se na inspiração deste trabalho numa linha de produção. Embora este projeto não tenha como foco a manipulação de frutos em movimento, o seu objetivo é servir de base para trabalhos futuros que se foquem nessa tarefa. Assim, a área retangular selecionada assemelha-se a uma secção de um tapete industrial na qual o braço robótico irá operar.

Capítulo 5

Conclusões

Neste capítulo são descritas as conclusões, gerais e específicas, desta dissertação, sendo discutidos todos os pontos fundamentais deste projeto. São também propostos projetos de trabalhos futuros que podem agregar todos os conceitos desenvolvidos por esta dissertação.

5.1 Conclusões gerais

Esta dissertação analisa o desenvolvimento e a aplicação de tecnologias avançadas no setor de alimentos, com foco específico na visão computacional e no uso da robótica. O problema central abordado foi a necessidade de melhorar continuamente a automação e a precisão nos processos da indústria alimentar, utilizando tecnologias não destrutivas, de fácil implementação e com vantagens económicas. Os principais objetivos incluíram a implementação de sistemas de visão computacional para a deteção de frutos, focando-se especificamente em tomate e laranja.

As ferramentas utilizadas para a criação dos modelos de visão computacional mostraram-se sólidas e fiáveis, especialmente a combinação do TensorFlow com o OpenCV, permitindo a criação acessível de modelos e fornecendo ferramentas eficazes para a aplicação e avaliação dos mesmos.

O *hardware* utilizado revelou-se eficaz para a tarefa proposta. O Raspberry Pi 5 lidou bem com o modelo e o controlo do braço robótico, mantendo-se estável durante todo o processo de testes e apresentando uma quantidade aceitável de *frames* por segundo. O UR3e também apresentou resultados positivos, permitindo a criação de uma área de trabalho com alcance admissível e confiável. A comunicação entre os dois dispositivos foi constante e com um tempo de resposta muito baixo.

Dos quatro modelos de deteção e classificação criados, três apresentaram resultados satisfatórios e passíveis de utilização em situações reais: os modelos de deteção e classificação de tomate com 2 e 4 classes e o modelo de deteção e classificação de laranjas com 2 classes. O modelo de deteção e classificação de laranjas com 4 classes não atingiu resultados satisfatórios devido à natureza da base de dados utilizada no seu treino.

O sistema final apresentou uma área de trabalho fiável de 65.500 mm^2, o que se mostrou suficiente para operações similares a um ambiente industrial.

O modelo de deteção e classificação de laranjas com 4 classes inicialmente apresentou resultados exageradamente bons, mostrando índices próximos de 100% de precisão. No entanto, em alguns casos, aparecia o valor "-1", o que levantou suspeitas. Após converter o modelo para o formato TensorFlow Lite e realizar uma segunda avaliação com outra ferramenta, constatou-se que algumas classes não podiam ser lidas corretamente. Testes com imagens não utilizadas no treino destacaram ainda mais essa falha, evidenciando a importância de usar múltiplos métodos de avaliação para validar um modelo. Outro ponto importante é a qualidade da base de dados: o modelo foi treinado com imagens tiradas no mesmo ambiente, todas centralizadas e com pouca variação entre as classes. Além de uma quantidade limitada de dados, o que não favorece a aprendizagem e gera falsos positivos na avaliação.

Outro aspeto abordado nesta dissertação é a importância de definir uma área de segurança ao manipular um braço robótico com cinemática inversa. Com esse método, o caminho seguido pelo braço pode variar, sendo crucial que toda a área de trabalho seja testada várias vezes para garantir a integridade do equipamento.

Os resultados obtidos evidenciam a eficácia das ferramentas e do *hardware* utilizados. A combinação do TensorFlow com o OpenCV provou ser uma escolha robusta para o desenvolvimento dos modelos de visão computacional, oferecendo um conjunto de ferramentas eficazes para a sua aplicação e avaliação. O Raspberry Pi 5 demonstrou capacidade suficiente para lidar com os modelos e controlar o braço robótico, enquanto o UR3e garantiu uma operação confiável e eficiente, com comunicação constante e tempos de resposta baixos.

Embora três dos quatro modelos desenvolvidos tenham apresentado resultados satisfatórios, o modelo de deteção e classificação de laranjas com 4 classes destacou a importância de uma base de dados de alta qualidade e da validação rigorosa dos modelos. Este caso específico sublinhou a necessidade de múltiplos métodos de avaliação e de um conjunto de dados diversificado e amplo para evitar falsos positivos e garantir a precisão dos modelos.

Além disso, a manipulação segura de braços robóticos utilizando cinemática inversa foi outro aspeto crucial abordado. A definição e o teste rigoroso da área de trabalho são fundamentais para assegurar a integridade do equipamento e a segurança operacional.

5.3 Sugestões de trabalhos futuros

Para a realização de trabalhos futuros, sugere-se a utilização de outros modelos de robôs, especialmente robôs delta, com o objetivo de estudar a velocidade de atuação em comparação com um braço robótico. Outra sugestão é a criação de uma base de dados própria, feita no ambiente de trabalho, para otimizar o modelo treinado com imagens típicas da sua utilização. Uma possível melhoria deste projeto é a implementação de uma forma para o robô conseguir apanhar os frutos em movimento, elevando a ideia deste projeto para um nível mais próximo da aplicabilidade real em indústria. Por fim, a criação de uma base de dados na nuvem que comunique diretamente com o programa permitirá averiguar a fiabilidade de cada carga recebida, catalogando a quantidade de frutas defeituosas. Desta forma, otimiza-se a produção e a seleção de fornecedores no futuro.

Bibliografia

[1] M. Völter, T. Stahl, J. Bettin, A. Haase e S. Helsen, Model-driven software development: technology, engineering, management, John Wiley & Sons., 2013.

[2] K. Schwab, The Fourth Industrial Revolution, WEF, 2016.

[3] H. MacKenzie, "http://www.belden.com," 07 10 2023. [Online]. Available: http://www.belden.com/blog/industrialethernet/The-Smart-Factory-of-theFuture-Part-1.cfm.

[4] R. Md. Sazzadur, G. Tapotosh, F. A. Nahid, S. K. M e A. Mehrin, "Machine learning and internet of things in industry 4.0: A review," *Measurement: Sensors,* 2023.

[5] T. F. a. A. Organization, "Global Initiative on Food Loss and Waste Reduction," [Online]. Available: https://www.fao.org/3/i4068e/i4068e.pdf. [Acedido em 20 Maio 2024].

[6] T. F. a. A. Organization, "How to Feed the World in 2050," [Online]. Available: https://www.fao.org/fileadmin/templates/wsfs/docs/expert_paper/How_to_Feed_the_World_in_2050.pdf. [Acedido em 20 Maio 2024].

[7] I. Brasil e M. Siddiqui, "Postharvest quality of fruits and vegetables: An overview," *Preharvest Modulation of Postharvest Fruit and Vegetable Quality,* pp. 1-40, 2018.

[8] P. Michela, C. Maria, P. Bernado, A. Giovanni e C. Giancarlo, "Computer vision system based on conventional imaging for non-destructively evaluating quality attributes in fresh and packaged fruit and vegetables," *Postharvest Biology and Technology* , vol. 200, 2023.

[9] M. Amodio, A. Cabezas-Serrano, R. Rinaldi e G. Colelli, "Rating the Various ADHD Rating Scales for Diagnostic Accuracy," *Pediatric Collections: ADHD: Evaluation and Care,* 2019.

[10] V. Narendra e V. Amithkumar, "An intelligent computer vision system for vegetables and fruits quality inspection using soft computing techniques," *Agric. Eng.,* vol. 3, nº 21, pp. 171-178, 2019.

[11] E. Maarten e S. Florian, "Global Demand for Food Is Rising. Can We Meet It?," Havard Business Review, 07 Abril 2016. [Online]. Available: https://hbr.org/2016/04/global-demand-for-food-is-rising-can-we-meet-it. [Acedido em 08 02 2024].

[12] A. R. Miguel, Fostering Innovation for Agriculture 4.0, A Comprehensive Plant Germplasm System, Cham, Switzerland: Springer, 2019.

[13] F. Y. Narvaez, G. Reina, M. Torres-Torriti, G. Kantor e F. A. Cheein, "A survey of ranging and imaging techniques for precision agriculture phenotyping," *IEEE/ASME Trans. Mechatron,* vol. 22, nº 6, pp. 2428-2439, 2027.

[14] E. Oztemel e S. Gursev, "Literature review of Industry 4.0 and related technologies," *J. Intell. Manuf,* vol. 31, nº 1, pp. 127-182, 2020.

[15] C. M. De, A. Vats e A. Biel, "Agriculture 4.0: The future of farming technology," *n Proc. World Government Summit,* pp. 11-13, 2018.

[16] L. Ye, M. Xiaoyuan, S. Lei, P. H. Gerhard e M. A.-M. Adnan, "From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research

Challenges," *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, vol. 17, nº 6, pp. 4322-4334, 2017.

[17] H. Heriansyah, I. I e N. A. , "Optimization of Herbal Dryer System Based on Smart Fuzzy and Internet of Thing (IOT)," *International Journal of Engineering and Science Applications,* vol. 6, nº 2, 2019.

[18] R. Badarinath e V. V. Prabhu, "Advances in Internet of things (IoT) in manufacturing," em *International Conference, APMS* , Hamburg, Germany,, 2017.

[19] L. D. Xu, W. He e S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics,* vol. 10, nº 4, pp. 2233 - 2243, 2014.

[20] E. Theodoridis, G. Mylonas e I. Chatzigiannakis, "Developing an IoT smart city framework," em *4th international Conference on information, intelligence, systems and applications*, Piraeus, Greece, 2014.

[21] Y. Fan, Y. J., Y. H., L. D. Xu, Y. Zeng e F. Wu, "IoT-based smart rehabilitation system," *IEEE Transactions on Industrial Informatics,* vol. 10, nº 2, pp. 1568-1577, 2014.

[22] Z. Pang, L. Zheng, J. Tian, S. Kao-Walter, E. Dubrova e Q. Chen, "Design of a terminal solution for integration of in-home health care devices and services towards the Internet-of-Things," *Enterprise Information Systems,* vol. 9, nº 1, pp. 86-116, 2015.

[23] J. Wang, H. Wang, J. He, L. Li, S. M. X. Tan e L. Zheng, "Wireless sensor network for real-time perishable food supply chain management," *Computers and Electronics in Agriculture*, vol. 110, nº 1, pp. 196-207, 2015.

[24] N. Chamara, D. Islam, G. Frank, Y. Shi e Y. Ge, "Ag-IoT for crop and environment monitoring : Past, present, and future," *Agricultural Systems,* vol. 203, 2022.

[25] M. Juandi, R. Joko e Gimin, "Drying fresh cassava chip using biomass energy with IoT monitoring system," *CIGR Journal,* vol. 24, nº 3, 2022.

[26] F. Civerchia, S. Bocchino, C. Salvadori, E. Rossi, L. Maggiani e M. Petracca, "Industrial Internet of Things monitoring solution for advanced predictive maintenance applications," *Journal of Industrial Information Integration,* vol. 7, nº 1, pp. 4-12, 2017.

[27] G. Heo, R. Manivannan, H. Kim, M. J. Kim, K. S. Min e Y. A. Son, "Developing an RGB - Arduino device for the multi-color recognition, detection and determination of Fe(III), Co(II), Hg(II) and Sn(II) in aqueous media by a terpyridine moiety.," *Sensors and Actuators, B: Chemical,* vol. 297, 2019.

[28] D. Harsh, P. K. Nema e K. A. Vinkel, "Internet of Things in food processing and its potential in Industry 4.0 era: A review," *Trends in Food Science & Technology,* vol. 139, nº 1, 2023.

[29] I. Jamshed, U. I. Raza, Z. ,. Syed, A. ,. Abdul e A. A. Syed, "AUTOMATING INDUSTRIAL TASKS THROUGH MECHATRONIC SYSTEMS – A REVIEW OF ROBOTICS IN INDUSTRIAL PERSPECTIVE," *Automatzacija industrijskih poslova kroz mehatroničke sustave – pregled robotike iz industrijske perspektive,* vol. 23, 2016.

[30] J. Iqbal, A. Mehmood, S. A. Alharbi, W. Alam, K. Ali e U. Javaid, "Nonlinear control of a flexible joint robotic manipulator with experimental validation," *Journal of Mechanical Engineering,* vol. 64, nº 1, pp. 47-55, 2018.

[31] P. Y. Chua, T. Ilschner e Caldwell, "Robotic manipulation of food products - A review," *Industrial Robot: An International Journal,* vol. 30, pp. 345-354, 2003.

[32] J. Gray e S. Davis, "Robotics in the food industry: an introduction," *Robotics and Automation in the Food Industry Current and Future Technologies,* pp. 21-35, 2013.

[33] B. Torgny, "Robot Control Overview: An Industrial Perspective," *Modelling Identification and Control,* vol. 30, nº 1, pp. 167-180, 2009.

[34] A. A. Syed, I. Jamshed, I. ,. Muhammad e M. Adeel, "A systematic review of current and emergent manipulator control approaches," *Frontiers of Mechanical Engineering,* vol. 10, nº 1, pp. 198-210, 2015.

[35] F. Lin e R. D. Brandt, "An optimal control approach to robust control of robot manipulators," *IEEE Transactions on Robotics and Automation,* vol. 14, nº 1, pp. 69-77, 1998.

[36] F. K. Muhammad, u. ,. Raza e I. Jamshed, "Control strategies for robotic manipulator," *International Conference on Robotics and Artificial Intelligence,* vol. 97, pp. 132-145, 2012.

[37] L. B. Terrence, "PID Advances in Industrial Control," *IFAC Proceedings Volumes,* vol. 45, nº 3, pp. 22-28, 2012.

[38] M. I. Ullah, S. A. Ajwad, M. Irfan e J. Iqbal, "Non-linear control law for articulated serial manipulators: Simulation augmented with hardware implementation.," *Elektronika Ir Elektrotechnika,* vol. 22, nº 1, pp. 3-7, 2016.

[39] R. U. I. Muhammad, K. Qudrat e I. Jamshed, "Design and Comparison of Two Control Strategies for Multi-DOF Articulated Robotic Arm Manipulator," *Control Engineering and Applied Informatics,* vol. 16, nº 2, pp. 28-39, 2014.

[40] H. K. Zeashan, K. Azfar e I. Jamshed, "Towards realizing robotic potential in future intelligent food manufacturing systems," *Innovative Food Science and Emerging Technologies,* vol. 48, pp. 11-28, 2018.

[41] T. Gopaiah, C. Enda, M. Cronan e O. John, "State of the art review of Big Data and web-based Decision Support Systems (DSS) for food safety risk assessment with respect to climate change," *Trends in Food Science & Technology,* vol. 126, pp. 192-204, 2022.

[42] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system[Online]," 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf. [Acedido em 14 02 2024].

[43] P. Xiangzhen, Z. Zhiyao, W. Xiaoyi, L. Haisheng, X. Jiping e Z. Xin, "A review on blockchain smart contracts in the agri-food industry: Current state, application challenges and future trends," *Computers and Electronics in Agriculture,* vol. 208, 2023.

[44] D. Shao, C. Kombe e S. Saxena, "An ensemble design of a cash crops-warehouse receipt system (WRS) based on blockchain smart contracts," *Journal of Agribusiness in Developing and Emerging Economies,* 2022.

[45] X. H. Kuang, T. D. e Z. X. Zhou, "Applying big data to food safety risk monitoring," *Software engineering and information technology,* pp. 181-84, 2016.

[46] X. Weipeng, Z. Xianghan, L. Xiaoliang, L. Xiaowei e F. Xionghui, "Agricultural Product Traceability System Based on Blockchain Technology," *EEE International Conference on Big Data and Cloud Computing (BdCloud),* pp. 1266-1270, 2019.

[47] Z. Zheng, S. Xie, H. Dai, W. Chen, X. Chen, J. Weng e M. Imran, "An overview on smart contracts: Challenges, advances and platforms.," *Futur. Gener. Comput. Syst,* vol. 105, pp. 475-491, 2020.

[48] S. Umamaheswari, S. Sreeram, N. Kritika e D. Prasanth, "Biot: blockchain based IoT for agriculture," em *11th International conference on advanced computing (ICoAC). IEEE*, 324-327, 2019.

[49] Q. Tao, X. Cui, X. Huang, A. Leigh e H. Gu, "Food safety supervision system based on hierarchical multi-domain blockchain network.," *IEEE Access 7,* vol. 7, p. 51817–51826, 2019.

[50] M. Osmanoglu, B. Tugrul, T. Dogantuna e E. Bostanci, "An effective yield estimation system based on blockchain technology," *IEEE Trans. Eng. Manag.,* vol. 67, nº 4, pp. 1157-1168, 2020.

[51] N. Yiu, "Decentralizing supply chain anti-counterfeiting and traceability systems using blockchain technology," *Future Internet,* vol. 13, nº 4, p. 84, 2021.

[52] J. Qian, Q. Yu, L. Jiang, H. Yang e W. Wu, "Food cold chain management improvement: A conjoint analysis on COVID-19 and food cold chain systems.," *Food Control,* vol. 137, 2022.

[53] M. Majdalawieh, N. Nizamuddin, M. Alaraj, S. Khan e A. Bani-Hani, "Blockchain based solution for Secure and Transparent Food Supply Chain Network," *Peer-to-Peer Network. Appl. ,* vol. 14, nº 6, pp. 3831-3850, 2021.

[54] O. Suhaili, R. M. Nidhi, H. M.A., A. ,. Norliza e M. ,. Jarinah, "Artificial intelligence-based techniques for adulteration and defect detections in food and agricultural industry: A review," *Journal of Agriculture and Food Research,* vol. 12, 2023.

[55] C. H. Lim, L. Steven, S. H. Bing, P. Q. N. Wendy, L. N. Sue, D. ,. Wei e L. ,. Hon, "A review of industry 4.0 revolution potential in a sustainable and renewable palm oil industry: HAZOP approach," *Journal of Agriculture and Food Research,* vol. 12, 2023.

[56] S. Hernandez e J.L. Lopez, "Uncertainty quantification for plant disease detection using Bayesian deep learning," *Appl. Soft Comput. J. ,* vol. 96, 2020.

[57] J. Han, M. Zuo, W. Zhu, J. Zuo, E. Lü e X. Yang, "A comprehensive review of cold chain logistics for fresh agricultural products : current status, challenges , and future trends," *Trends Food Sci. Technol,* vol. 109, pp. 536-551, 2021.

[58] A. Mellit e S. Kalogirou, "Artificial intelligence techniques for photovoltaic applications," *Progress in Energy and Combustion Science,* vol. 34, nº 5, pp. 574-632, 2008.

[59] D. Saha e A. Manickavasagan, "Machine learning techniques for analysis of hyperspectral images to determine quality of food products : a review," *Curr. Res. Food Sci,* vol. 4, pp. 28-44, 2021.

[60] K. Liakos, P. Busato, D. M. S. Pearson e D. Bochtis, "Machine learning in agriculture: a review," *Sensors,* vol. 18, nº 8, pp. 1-29, 2018.

[61] Y. Yang, Z. Liu, M. Huang, Q. Zhu e X. Zhao, "Automatic detection of multi-type defects on potatoes using multispectral imaging combined with a deep learning model," *J. Food Eng.,* vol. 336, 2023.

[62] N. Kunhare, R. Tiwari e J. Dhar, "Intrusion detection system using hybrid classifiers with meta-heuristic algorithms for the optimization and feature selection," *genetic algorithm, Comput. Electr. Eng,* vol. 103, p. 108383, 2022.

[63] J. J. Geng, X. Liu, B. S. Kong e Z. Niu, “The fishmeal adulteration identification based on microscopic image and deep learning,” *Comput. Electron. Agric.,* vol. 198, 2022.

[64] J. Brownlee, “Deep Learning for Computer Vision: Image Classification, Object Detection and Face Recognition in Python,” em *Deep Learning for Computer Vision: Image Classification, Object Detection and Face Recognition in Python* , 2020, pp. 4-7.

[65] B. Jason, “MachineLearningMastery.com,” Guiding Tech Media, 05 Julho 2019. [Online]. Available: https://machinelearningmastery.com/what-is-computer-vision/. [Acedido em 14 Fevereiro 2024].

[66] Babatunde, O. Hezekiah, L. Armstrong, J. Leng e D. Diepeveen., “A survey of computer-based vision systems for automatic identification of plant species.,” *Journal of Agricultural Informatics,* pp. 61-71, 2015.

[67] O. Cosido, I. Andres, G. Akemi, C. Raffaele, C. Massimiliano, T. Leticia e S. Esteban, “Hybridization of Convergent Photogrammetry, Computer Vision, and Artificial Intelligence for Digital Documentation of Cultural Heritage-A Case Study: The Magdalena Palace,” *Cyberworlds (CW),International Conference IEEE,* pp. 369-379, 2014.

[68] K. K. Patel, A. Kar, S. N. Jha e M. A. Khan, “"Machine vision system: a tool for quality inspection of food and agricultural products.,” *Journal of food science and technology 49,* pp. 123-141, 2012.

[69] Rautaray, S. Siddharth e A. Anupam, “Vision-based hand gesture recognition for human-computer interaction: a survey.,” *Artificial Intelligence Review 43,* pp. 1-54, 2015.

[70] S. Ullman, A. Liav, F. Ethan e H. Daniel, “Atoms of recognition in human and computer vision,” *Proceedings of the National Academy of Sciences 113,* pp. 2744-2749, 2016.

[71] F. Zhao, X. Xie e M. Roach, “Computer Vision Techniques for Transcatheter Intervention,” *IEEE Journal of Translational Engineering in Health and Medicine.,* 2015.

[72] E. Mohamed, “How Does Computer Vision Work?,” 13 Fevereiro 2019. [Online]. Available: https://freecontent.manning.com/mental-model-graphic-grokking-deep-learning-for-computer-vision/. [Acedido em 20 Janeiro 2024].

[73] M. Sigdel, I. Dinc, M. Sigdel, S. Dinc, M. Pusey e R. Aygun, “Feature analysis for classification of trace fluorescent labeled protein crystallization images.,” *BioData Mining,* 2017.

[74] S. Matiacevich, C. Celis, S. D, E. P e O. J, “Quality Parameters of Six Cultivars of Blueberry Using Computer Vision.,” *International Journal of Food Science,* 2013.

[75] C. Junyi, Z. Hao, L. Anming, W. Eric e N. T, “Deep learning in computer vision: A critical review of emerging techniques and application scenarios,” *Machine Learning with Applications 6,* 2021.

[76] P. Viola e M. Jones, “Rapid object detection using a boosted cascade of simple features,” em *Computer society conference on computer vision and pattern recognition,* USA, 2001.

[77] P. Jones, Viola e M. J., “Robust real-time face detection,” *International Journal of Computer Vision,* pp. 137-154, 2004.

[78] "OpenCV," Google, [Online]. Available: https://docs.opencv.org/4.x/d2/d99/tutorial_js_face_detection.html. [Acedido em 26 Janeiro 2024].

[79] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, W. S. e S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing,* p. 27–48, 2016.

[80] J. Redmon, S. Divvala, R. Girshick e A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 779-788, 2016.

[81] A. Krizhevsky, I. Sutskever e G. Hinton, "Imagenet classification with deep convolutional neural networks," *In Proceedings of the 25th international conference on neural information processing systems,* pp. 1097-1105, 2012.

[82] K. Simonyan e A. Zisserman, "Very deep convolutional networks for large- scale image recognition," *In Proceedings of the 3rd international conference on learning representations,* 2015.

[83] M. Lin, Q. Chen e S. Yan, "Network in network," *In Proceedings of the 2014 international conference on learning representations,* 2014.

[84] K. He, X. Zhang, S. Ren e J. Sun, "Deep residual learning for image recognition resnet," *In Proceedings of the 2016 ieee conference on computer vision and pattern recognition,* pp. 770-778, 2016.

[85] G. Huang, Z. Liu, L. van der Maaten e K. Q. Weinberger, "Densely connected convolutional networks," *In Proceedings of the 2017 ieee conference on computer vision and pattern recognition,* pp. 2261-2269, 2017.

[86] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto e H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv,* 2017.

[87] M. Tan e Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *In Proceedings of the 36thinternational conference on machine learning,* pp. 6105-6114, 2019.

[88] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He e P. Dollár, "Designing network design spaces," *In Proceedings of the 2020 ieee/cvf conference on computer vision and pattern recognition,* pp. 10425-10433, 2020.

[89] Z. Zou, K. Chen, Z. Shi, Y. Guo e J. Ye, "Object detection in 20 years: A survey," *IEEE,* 2023.

[90] S. Hao, Y. Zhou e Y. Guo, "A brief survey on semantic segmentation with deep learning," *Neurocomputing 406,* pp. 302-231, 2020.

[91] H. Yuning, Q. Yurong, W. Hongyang, L. Yiguo, L. Bowen e Q. Yugang, "A survey of deep learning-based object detection methods in crop counting," *Computers and Electronics in Agriculture,* 2023.

[92] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu e A. Berg, "Ssd: Single shot multibox detector," *Computer Vision–ECCV 2016: 14th European Conference,* pp. 21-37, 2016.

[93] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov e S. Zagoruyko, "End-to-end object detection with transformers.," *Computer Vision–ECCV 2020: 16th European Conference,* pp. 213-229, 2020.

[94] J. Redmon e F. A. , "YOLO9000: better, faster, stronger," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 7263-7271, 2017.

[95] J. Redmon e A. Farhad, "Yolov3: An incremental improvement," *University of Washington,* 2018.

[96] A. Bochkovskiy, C.-Y. Wang e H.-Y. Liao, "Yolov4: Optimal speed and accuracy of object detection," *Cornell University, Computer Vision and Pattern Recognition,* 2020.

[97] J. Glenn, C. Ayush, S. Alex, B. Jirka, K. Yonghye, M. Kalen, TaoXie, F. Jiacong, Y. Zeng, W. Colin, Abhiram, M. Diego, W. Zhiqiang, F. Cristi, N. Jebastin, S. Victor, S. Piotr, H. Adam, N. Dhruv e S. Max, "Zenodo," 22 Novembro 2022. [Online]. Available: https://zenodo.org/records/7347926. [Acedido em 26 Janeiro 2024].

[98] G. Zheng, L. Songtao, W. Feng, L. Zeming e S. Jian, "YOLOX: Exceeding YOLO Series in 2021," *Conerll University, Computer Vision and Pattern Recognition,* 2021.

[99] L. Chuyi, L. Lulu, J. Hongliang, W. Kaiheng, G. Yifei, L. Liang, K. Zaidan, L. Qingyuan, C. Meng, N. Weiqiang, L. Yiduo, Z. Bo, L. Yufei, Z. Linyuan, X. Xiaoming, C. Xiangxiang, W. Xiaoming e W. Xiaolin, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," *Cornell University Computer Vision and Pattern Recognition,* 2022.

[100] C.-Y. Wang, A. Bochkovskiy e H.-Y. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.," *Cornell University, Computer Vision and Pattern Recognition,* 2022.

[101] G. Jocher, A. Chaurasia e J. Qiu, "GitHub," Ultralytics, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics?tab=readme-ov-file. [Acedido em 26 Janeiro 2024].

[102] A. Hyochang e H.-J. Cho, "Research of automatic recognition of car license plates based on deep learning for convergence traffic control system," *Personal and Ubiquitous Computing ,* pp. 1139-1148, 2023.

[103] R. Girshick, J. Donahue, T. Darrell e J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans,* vol. 38, pp. 142-158, 2015.

[104] K. He, X. Zhang, S. Ren e J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans,* vol. 37, pp. 1904-1916, 2015.

[105] S. W. Gurjit e K. Rajiv, "Recent advances on multicue object tracking: a survey," 09 Janeiro 2016. [Online]. Available: https://doi.org/10.1007/s10462-015-9454-6. [Acedido em 26 Janeiro 2024].

[106] A. Yilmaz, O. Javed e M. Shah, "Object tracking: a survey," *ACM Comput Surv 38,* vol. 38, pp. 1-45, 2006.

[107] S. Yun, J. Choi, Y. Yoo, K. Yun e C. J. Y. , "Action-decision networks for visual tracking with deep reinforcement learning," *Proceedings of the 2017 ieee conference on computer vision and pattern recognition,* pp. 1349-1358, 2017.

[108] L. Alan, M. Jiri e K. Matej, "D3S – A Discriminative Single Shot Segmentation Tracker," *IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2020.

[109] J. Gilles, "Empirical wavelet transform," *IEEE Transactions on Signal Processing,* vol. 61, nº 16, pp. 3999-4010, 2013.

[110] G. M, R. I, P. G e Z. P, “Discrete wavelets transform for improving greenness image segmentation in agricultural images,” *Computers and Electronics in Agriculture,* vol. 118, pp. 396-407, 2015.

[111] L. Zifei, Y. Wenzhu, Y. Yunfeng, G. Ruru e L. Xiaonan, “Semantic segmentation of agricultural images: A survey,” *Information Processing in Agriculture,* 2023.

[112] B.-E. Fredy, G. Esther, I. L.-G. Clara, G.-S. María J. e P. Gonzola, “Semantic segmentation based on Deep learning for the detection of Cyanobacterial Harmful Algal Blooms (CyanoHABs) using synthetic images,” *Applied Soft Computing,* vol. 141, 2023.

[113] O. Ronneberger, P. Fischer e T. Brox, “ T. U-Net: Convolutional networks for biomedical image segmentation,” *International Conference on Medical image computing and computerassisted intervention,* pp. 234-241, 2015.

[114] Z. Kunlin, C. Xin, W. Yonglin e Z. Chunlong, “A modified U-Net with a specific data argumentation method for semantic segmentation of weed images in the field,” *Computers and Electronics in Agriculture,* nº 187, 2021.

[115] C. Liang-Chieh, P. George, K. Iasonas, M. Kevin e Y. Alan L, “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 40, nº 4, pp. 834-848, 2018.

[116] M. Chen, C. Wang, B. Fei, B. Fei, B. Zhang, S. Zhang e A. Huang, “Biological degradation of chinese fir with trametes versicolor (l.) lloyd,” *IEEE Trans Pattern Anal Mach,* vol. 4, nº 40, pp. 834-848, 2017.

[117] C. Liang-Chieh, P. G, S. Florian e A. Hartwig, “Rethinking Atrous Convolution for Semantic Image Segmentation,” *Proceedings of the European Conference on Computer Vision,* pp. 801-818, 2018.

[118] H. Kaiming, Z. X, R. Shaoqing e S. Jian, “Identity Mappings in Deep Residual Networks,” *European Conference on Computer Vision,* 2016.

[119] L. Chen, Y. Zhu, G. Papandreou, F. Schroff e H. Adam, “ EncoderDecoder with atrous separable convolution for semantic image segmentation,” em *Proceedings of the European conference on computer vision. ,* 2018.

[120] A. Ahmad e A. Lujain, “Monitoring deforestation in Jordan using deep semantic segmentation with satellite imagery,” *Ecological Informatics,* vol. 70, 2022.

[121] S. Kaiqiong, W. Xuan, L. Shoushuai e L. ChangHua, “Apple, peach, and pear flower detection using semantic segmentation network and shape constraint level set,” *Computers and Electronics in Agriculture,* vol. 185, 2021.

[122] J. Yongwon, L. Soobin, L. Youngjae, K. Hyungu, P. Seonghun, B. Seounghun, K. Minkwan, H. Sungwon e K. Seoungbum, “Semantic Segmentation of Cabbage in the South Korea Highlands with Images by Unmanned Aerial Vehicles,” *Applied Sciences ,* vol. 11, 2021.

[123] W. Aamir, N. Asma, T. Maria e S. Gilani, “Recent progress in digital image restoration techniques: A review,” *Digital Signal Processing,* vol. 141, pp. 104187-104203, 2023.

[124] K. A, S. I e H. G.E, “ImageNet classification with deep convolutional neural networks,” em *n Proceedings of the 25th International Conference on Neural Information Processing,* Lake Tahoe, Nevada, USA , 2012.

[125] C. K, V. M e K. S, “Semi-supervised learning for low-dose ct image restoration with hierarchical deep generative adversarial network (hd-gan),” em *41st Annual*

International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE, 2019.

[126] A. Daneshmand, F. Facchinei, V. Kungurtsev e G. Scutari, "Flexible selective parallel algorithms for big data optimization," em *Proceedings of the 48th Asilomar Conference on Signals, Systems and Computers*, 2014.

[127] A. Daneshmand, F. Facchinei, V. Kungurtsev e G. Scutari, "Hybrid random/deterministic parallel algorithms for convex and nonconvex big data optimization.," *IEEE Trans. Signal Process.*, vol. 15, nº 63, p. 3914–3929, 2015.

[128] H. Zhu, K. Yuen, L. Mihaylova e H. Leung, "Overview of environment perception for intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, nº 18, p. 2584–2601, 2017.

[129] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 2, nº 60, pp. 91-110, 2004.

[130] C. Ruilong, M. Lyudmila, Z. Hao e C. B. Nidhal, "A Deep Learning Framework for Joint Image Restoration and," *Circuits Syst. Signal Process*, nº 39, p. 1561–1580, 2020.

[131] A. Krull, T.-O. Buchholz e F. Jug, "Noise2Void—LEarning denoising from single noisy images," em *n Proceedings of the 2019 ieee/cvf conference on computer vision and pattern recognition*, USA, 2019.

[132] W. Z. Syed, A. Aditya, K. Salman, H. Munawar, S. K. Fahad, Y. Ming-Hsuan e S. Ling, "Cycleisp: Real image restoration via improved data synthesis," em *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[133] D. Lorente, N. Aleixos, J. Gomez-Sanchis, S. Cubero e O. García-Navarrete, "Recent advances and applications of hyperspectral imaging for fruit and vegetable quality assessment," *Food Bioproc. Tech.*, vol. 5, nº 4, pp. 1121-1142, 2012.

[134] D. Lorente, N. Aleixos, J. Gomez-Sanchis, S. Cubero e O. García-Navarrete, "Recent advances and applications of hyperspectral imaging for fruit and vegetable quality assessment," *Food Bioproc. Tech.*, vol. 5, nº 4, pp. 1121-1142, 2012.

[135] K. Sendin, M. Manley e P. Williams, "Classification of white maize defects with multispectral imaging," *Food Chem*, vol. 243, nº 1, pp. 311-318, 2018.

[136] G. Leiva-Valenzuela e J. Aguilera, "Automatic detection of orientation and diseases in blueberries using image analysis to improve their postharvest storage quality," *Food Control*, vol. 33, nº 1, p. 166–173, 2013.

[137] Y. Akter e M. Rahman, "Development of a computer vision based eggplant grading system," em *4th International Conference on Advances in Electrical Engineering (ICAEE)*, Dhaka, Bangladesh, 2017.

[138] X. Huang, S. Yu, H. Xu, J. Aheto, E. Bonah, M. Ma, M. Wu e X. Zhang, "Rapid and nondestructive detection of freshness quality of postharvest spinaches based on machine vision and electronic nose," *J. Food Saf.* , vol. 39, nº 6, 2019.

[139] Y. Chen, J. Wu e M. Cui, "Automatic classification and detection of oranges based on computer vision," em *4th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2018.

[140] P. Moallem, A. Serajoddin e H. Pourghassem, “Computer vision-based apple grading for golden delicious apples based on surface features,” *Inf. Process. Agric,* vol. 4, nº 1, pp. 33-40, 2017.

[141] D. Sahu e R. Potdar, “Defect identification and maturity detection of mango fruits using image analysis,” *Am. J. Artif. Intell,* vol. 1, nº 1, pp. 5-14, 2017.

[142] M. Arakeri e L. P., “Computer vision based fruit grading system for quality evaluation of tomato in agriculture industry.,” *Procedia Comput.,* vol. 79, nº 1, pp. 426-433, 2016.

[143] H. Kaiming, Z. Xiangyu, R. Shaoqing e S. Jian, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” em *EEE International Conference on Computer Vision (ICCV),* Santiago, Chile, 2015.

[144] Y. F. J. W. Z. L. Longsheng Fu, G. Fangfang, M. Yaqoob, A.-M. Ahmad, Z. Qin, L. Rui e C. Yongjie, “Fast and accurate detection of kiwifruit in orchard using improved YOLOv3-tiny model,” *Precision Agric,* vol. 22, pp. 754-776, 2021.

[145] B. ,. Raihan, A. Samiha, I. Faria, S. ,. ,. Md e K. Riasat, “Deep learning based object detection and surrounding environment description for visually impaired people,” *Heliyon,* vol. 9, nº 6, 2023.

[146] S. Mark, H. Andrew, Z. Menglong e Z. Andrey, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” em *Conference on Computer Vision and Pattern Recognition (CVPR),* Salt Lake City, UT, USA, 2018.

[147] A. Pratiksha e R. Sayali, “Pick and place industrial robot controller with computer vision,” em *International Conference on Computing Communication Control and automation (ICCUBEA),* Pune, India, 2016.

[148] G. Rahul, G. Ankush, Z. Ashish e M. Ajay, “Review on Development of Industrial Robotic Arm,” *International Research Journal of Engineering and Technology (IRJET),* vol. 4, nº 3, 2017.

[149] M. Spong, S. Hutchinson e M. Vidyasagar, “Robot Modeling and Control,” Hoboken, John Wiley and Sons, 2006.

[150] S. Bruno e K. Oussama, “Springer Handbook of Robotics,” Berlin, Springer, 2016.

[151] J. Denavit e R. Hartenberg, “A kinematic notation for lower-pair mechanisms based on matrices,” *ASME Journal of Applied Mechanics,* vol. 22, pp. 215-221, 1955.

[152] B. Siciliano, L. Sciavicco, L. Villani e G. Oriolo, “ Robotics: modelling, planning and control, advanced textbooks in control and signal processing,” London, UK, Springer, 2009.

[153] “UR3E ultra-lightweight, Compact Cobot,” Universal Robots, [Online]. Available: https://www.universal-robots.com/products/ur3-robot/. [Acedido em 2024 Fevereiro 20].

[154] “Universal Robots - user manual - UR3E e-series - SW 5.7 - english us (EN-US),” Collaborative robotic automation, [Online]. Available: https://www.universal-robots.com/download/manuals-e-series/user/ur3e/57/user-manual-ur3e-e-series-sw-57-english-us-en-us/. [Acedido em 20 Fevereiro 2024].

[155] “Universal robots - DH parameters for calculations of kinematics and dynamics,” Universal Robots, [Online]. Available: https://www.universal-

robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/. [Acedido em 20 Fevereiro 2024].

[156] R. Chen, R. Song, Z. Zhang, L. Bai, F. Liu, P. Jiang, D. Sindersberger, G. J. Monkman e J. Guo, "Bio-Inspired Shape-Adaptive Soft Robotic Grippers Augmented with Electroadhesion Functionality," *Soft Robotics,* p. 13, 2019.

[157] J. Yao, Y. Fang, X. Yang, P. Wang e L. Li, "Design optimization of soft robotic fingers biologically inspired by the fin ray effect with intrinsic force sensing," *Mechanism and Machine Theory,* p. 27, 2024.

[158] B. Simone, "Raspberry Pi reaches production of a million monthly unit," Verdict, 05 Junho 2023. [Online]. Available: https://www.verdict.co.uk/raspberry-pi-reaches-production-of-a-million-monthly-units/. [Acedido em 21 Fevereiro 2024].

[159] K. Stan e W. Chad, "Raspberry Pi as a Platform for the Internet of Things Projects: Experiences and Lessons," *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education,* 2017.

[160] H. Gareth, "Raspberry Pi Beginner's Guide 4th Edition," Cambridge, UK, Raspberry Pi Press, 2020.

[161] "Raspberry pi documentation - raspberry pi 5," Raspberry pi Foundation, [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html. [Acedido em 21 Fevereiro 2024].

[162] "Raspberry pi 5 mechanical drawing," [Online]. Available: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-mechanical-drawing.pdf. [Acedido em 21 Fevereiro 2024].

[163] "Raspberry pi camera algorithm and Tuning Guide," Raspberry pi Foundation , [Online]. Available: https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf. [Acedido em 21 Fevereiro 2024].

[164] "PYPL popularity of Programming Language index," [Online]. Available: https://pypl.github.io/PYPL.html. [Acedido em 22 Abril 2024].

[165] B. Gary, "The OpenCV Library," drdobbs, 01 Novembro 2000. [Online]. Available: https://www.drdobbs.com/open-source/the-opencv-library/184404319?pgno=1. [Acedido em 21 Fevereiro 2024].

[166] "Home," OpenCv, [Online]. Available: https://opencv.org/. [Acedido em 30 Maio 2024].

[167] P. Bo, N. Erik e N. W. Ying, "Deep Learning With TensorFlow: A Review," *Journal of Educational and Behavioral Statistics,* vol. 45, nº 2, pp. 227-248, 2019.

[168] O. Alsing, "Mobile Object Detection using TensorFlow Lite and Transfer Learning," 20118. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1242627&dswid=-8635. [Acedido em 22 Fevereiro 2024].

[169] B. Jason, "A Gentle Introduction to the Rectified Linear Unit (ReLU)," Machine Learning Mastery, 20 Agosto 2020. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. [Acedido em 22 Fevereiro 2024].

[170] S. Devin, "ReLU6: A Modified Version of Rectified Linear Unit," SERP AI, [Online]. Available: https://serp.ai/relu6/. [Acedido em 22 Fevereiro 2024].

[171] L. Zongshuai, X. Xuyu, Q. Jiaohua, T. Yun, Z. Qin e N. ,. Neal, "Image Recognition of Citrus Diseases Based on Deep Learning," *Computers, Materials & Continua,* vol. 66, nº 1, pp. 457-466, 2021.

[172] J. Evan, "TensorFlow Lite Object Detection Model Performance Comparison," 10 Dezembro 2022. [Online]. Available: https://www.ejtech.io/learn/tflite-object-detection-model-comparison. [Acedido em 22 Fevereiro 2024].

[173] Y. Morvan, "Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video," *Technische Universiteit Eindhoven,* 2009.

[174] "Camera Modeling: Exploring Distortion and Distortion Models, Part I," RSS, [Online]. Available: https://www.tangramvision.com/blog/camera-modeling-exploring-distortion-and-distortion-models-part-i. [Acedido em 06 Maio 2024].

[175] "Gunwerks LLC," [Online]. Available: https://www.gunwerks.com/blog/revic-11/optical-aberrations-in-rifle-scopes-287. [Acedido em 13 Maio 2024].

[176] H. Richard e Z. Andrew, Multiple View Geometry in Computer Vision Second Edition, Cambridge: Cambridge University Press, 2004.

[177] K. Simek, "A Computer Vision Blog," Sightations, 13 Agosto 2013. [Online]. Available: https://ksimek.github.io/2013/08/13/intrinsic/. [Acedido em 06 Maio 2024].

[178] R. Szeliski, Computer Vision: Algorithms and Applications 2nd, Cham: Springer, 2021.

[179] C. Cowan, B. Modayur e J. DeCurtins, "Automatic light-source placement for detecting object features," *Proceedings of SPIE,* vol. 1826, nº 1, pp. 397-408, 1992.

[180] S. K. Kopparapu, "Lighting design for machine vision application," *Image and Vision Computing,* vol. 1, nº 24, pp. 720-726, 2006.

[181] S. K. Kopparapu, "Lighting design for machine vision application," *Image and Vision Computing,* vol. 1, nº 24, pp. 720-726, 2006.

[182] J. Cristovão, "Kaggle," 7 Janeiro 2022. [Online]. Available: https://www.kaggle.com/datasets/jonathansilva2020/orange-diseases-dataset. [Acedido em 03 04 2024].

[183] "Kaggle," [Online]. Available: https://www.kaggle.com/. [Acedido em 2024 04 03].

[184] "Roboflow Universe," Roboflow, [Online]. Available: https://universe.roboflow.com/. [Acedido em 03 Abril 2024].

[185] J. Cartucho, R. Ventura e M. Veloso, "Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 2336-2341, 2018.

[186] C. Hong-Tae, L. Ho-Jun, K. Hoon e P. Ho-Hyun, "SSD-EMB: An Improved SSD Using Enhanced Feature Map Block for Object Detection," *mdpi Sensors,* vol. 2842, 2021.

[187] M. Simukayi, S. Shawn e H. Richard, "Understanding artificial intelligence based radiology studies: CNN architecture," *Clinical Imaging,* vol. 80, nº 72-76, pp. 72-76, 2021.

[188] L. Chen, Y. Zhu, G. Papandreou, F. Schroff e H. Adam, "EncoderDecoder with atrous separable convolution for semantic," *Proceedings of the European conference on computer vision,* pp. 801-818, 2018.

[189] “Buy A raspberry pi – raspberry pi,” Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.com/products/. [Acedido em 21 Fevereiro 2024].

[190] K. Simek, “A Computer Vision Blog,” Sightations, 13 Agosto 2013. [Online]. Available: https://ksimek.github.io/2013/08/13/intrinsic/. [Acedido em 06 Maio 2024].

[191] B. Anuja e B. Atul, “Fruits and vegetables quality evaluation using computer vision: A review,” *Journal of King Saud University - Computer and Information Sciences,* vol. 33, nº 3, pp. 243-257, 2021.

Anexo A Códigos utilizados

A.1 Código de controlo do UR3e com visão computacional

```
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util
from picamera2 import Picamera2
import socket
import struct
import math

HOST = "169.254.148.38"
PORT = 30002
PORT_DATA = 30003

class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self, resolution=(1280,720), framerate=30):
        self.picam2 = Picamera2()
        self.preview_config =
self.picam2.create_preview_configuration(main={"format": "XRGB8888", "size":
(1280,720)})
        self.picam2.configure(self.preview_config)
        self.frame = None
        self.stopped = False
        self.resolution = resolution

    def start(self):
        self.picam2.start()
        Thread(target=self.update, args=()).start()
        return self

    def update(self):
        while True:
            if self.stopped:
                return
            self.frame = self.picam2.capture_array()

    def read(self):
        return self.frame

    def stop(self):
        self.stopped = True
        self.picam2.stop()

# Define and parse input arguments
parser = argparse.ArgumentParser()
parser.add_argument('--modeldir', help='Folder the .tflite file is located
in',
                    required=True)
parser.add_argument('--graph', help='Name of the .tflite file, if different
than detect.tflite',
                    default='detect.tflite')
parser.add_argument('--labels', help='Name of the labelmap file, if different
than labelmap.txt',
                    default='labelmap.txt')
parser.add_argument('--threshold', help='Minimum confidence threshold for
displaying detected objects',
                    default=0.8)
```

```
parser.add_argument('--resolution', help='Desired webcam resolution in WxH. If
the webcam does not support the resolution entered, errors may occur.',
                    default='1280x720')
parser.add_argument('--edgetpu', help='Use Edge TPU', action='store_true')

args = parser.parse_args()

MODEL_NAME = args.modeldir
GRAPH_NAME = args.graph
LABELMAP_NAME = args.labels
min_conf_threshold = float(args.threshold)
resW, resH = args.resolution.split('x')
imW, imH = int(resW), int(resH)
use_TPU = args.edgetpu

pkg = importlib.util.find_spec('tflite_runtime')
if pkg:
    from tflite_runtime.interpreter import Interpreter
else:
    from tensorflow.lite.python.interpreter import Interpreter

# Get path to current working directory
CWD_PATH = os.getcwd()

# Path to .tflite file, which contains the model that is used for object
detection
PATH_TO_CKPT = os.path.join(CWD_PATH, MODEL_NAME, GRAPH_NAME)

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH, MODEL_NAME, LABELMAP_NAME)

# Load the label map
with open(PATH_TO_LABELS, 'r') as f:
    labels = [line.strip() for line in f.readlines()]

if labels[0] == '???':
    del(labels[0])

interpreter = Interpreter(model_path=PATH_TO_CKPT)
interpreter.allocate_tensors()

# Get model details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]

floating_model = (input_details[0]['dtype'] == np.float32)

input_mean = 127.5
input_std = 127.5

# Check output layer name to determine if this model was created with TF2 or
TF1,
# because outputs are ordered differently for TF2 and TF1 models
outname = output_details[0]['name']

if 'StatefulPartitionedCall' in outname:  # This is a TF2 model
    boxes_idx, classes_idx, scores_idx = 1, 3, 0
else:  # This is a TF1 model
    boxes_idx, classes_idx, scores_idx = 0, 1, 2

# Initialize frame rate calculation
frame_rate_calc = 1
freq = cv2.getTickFrequency()

# Initialize video stream
videostream = VideoStream(resolution=(imW, imH), framerate=30).start()
```

```
time.sleep(1)

def pixel_para_metros(x, y, resolution, fov_horizontal, fov_vertical,
ponto_referencia=(510,229), distancia_z=0.42):
    # Se ponto_referencia não for fornecido, usa o centro da imagem
    if ponto_referencia is None:
        ponto_referencia = (resolution[0] / 2, resolution[1] / 2)

    referencia_x, referencia_y = ponto_referencia

    # Calcula o deslocamento de pixels a partir do ponto de referência
    dx_pixel = x - referencia_x
    dy_pixel = y - referencia_y


    largura_metros = 2 * distancia_z * math.tan(math.radians(fov_horizontal /
2) )
    altura_metros = 2 * distancia_z * math.tan(math.radians(fov_vertical / 2)
)

    # Calcula quantos metros cada pixel representa
    metro_por_pixel_x = largura_metros / resolution[0]
    metro_por_pixel_y = altura_metros / resolution[1]

    # Converte o deslocamento de pixels para metros
    dist_y_metros = dx_pixel * metro_por_pixel_x *-1
    dist_x_metros = dy_pixel * metro_por_pixel_y

    return dist_x_metros, dist_y_metros
def enviar_comando_movimento_linear(x, y, z, rx, ry, rz, v=0.8, a=0.4, r=0):
    cmd = f"movej(p[{x}, {y}, {z}, {rx}, {ry}, {rz}], a={a}, v={v},
r={r})\n".encode('utf8')
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        s.send(cmd)
        time.sleep(2)  # Espera para o movimento ser completado

def enviar_comando_movimento_juntas(x, y, z, rx, ry, rz, v=0.2, a=0.3, r=0):
    cmd = f"movej([{x}, {y}, {z}, {rx}, {ry}, {rz}], a={a}, v={v},
r={r})\n".encode('utf8')
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        s.send(cmd)
        time.sleep(3)  # Espera para o movimento ser completado

def ler_posicao_atual():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT_DATA))
        time.sleep(1)  # Aguarda um momento para garantir que os dados estejam
sendo recebidos
        data = s.recv(1112)
        pos = struct.unpack('!6d', data[444:492])
        return pos

def desenhar_matriz(frame, linhas=10, colunas=10):
    altura, largura, _ = frame.shape
    for i in range(1, linhas):
        cv2.line(frame, (0, i * altura // linhas), (largura, i * altura //
linhas), (255, 255, 255), 1)
    for j in range(1, colunas):
        cv2.line(frame, (j * largura // colunas, 0), (j * largura // colunas,
altura), (255, 255, 255), 1)

opengripper = open('/home/senha123/Desktop/BibN/1win0in13uro/open.script',
'rb')  # Robotiq Gripper
closegripper = open('/home/senha123/Desktop/BibN/1win0in13uro/close.script',
'rb')  # Robotiq Gripper
```

```
activategripper =
open('/home/senha123/Desktop/BibN/1win0in13uro/Gripper.script', 'rb')  #
Robotiq Gripper

fov_horizontal = 66
fov_vertical = 41
resolution = (1280, 720)

# Ativar o gripper
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    a = activategripper.read(1024)
    while a:
        s.send(a)
        a = activategripper.read(1024)
    print("activate")
    time.sleep(8)
while True:
    t1 = cv2.getTickCount()
    frame1 = videostream.read()
    frame = frame1.copy()
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_resized = cv2.resize(frame_rgb, (width, height))
    input_data = np.expand_dims(frame_resized, axis=0)

    if floating_model:
        input_data = (np.float32(input_data) - input_mean) / input_std

    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()

    boxes = interpreter.get_tensor(output_details[boxes_idx]['index'])[0]
    classes = interpreter.get_tensor(output_details[classes_idx]['index'])[0]
    scores = interpreter.get_tensor(output_details[scores_idx]['index'])[0]

    num_objects_detected = 0

    for i in range(len(scores)):
        if scores[i] > min_conf_threshold and scores[i] <= 1.0:
            num_objects_detected += 1
            ymin, xmin, ymax, xmax = boxes[i]
            centroX, centroY = int((xmin + xmax) / 2 * imW), int((ymin + ymax)
/ 2 * imH)
            cv2.putText(frame, f"ID: {int(classes[i])}, Conf:
{scores[i]:.2f}", (centroX, centroY), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,
255, 0), 2, cv2.LINE_AA)
            cv2.rectangle(frame, (int(xmin * imW), int(ymin * imH)), (int(xmax
* imW), int(ymax * imH)), (10, 255, 0), 2)

    key = cv2.waitKey(1) & 0xFF
    if key == ord("p"):
        for i in range(len(scores)):
            if scores[i] > min_conf_threshold and scores[i] <= 1.0:
                print("entrou nesse if")
                ymin, xmin, ymax, xmax = boxes[i]
                centroX, centroY = int((xmin + xmax) / 2 * imW), int((ymin +
ymax) / 2 * imH)

                dist_x_metros, dist_y_metros = pixel_para_metros(centroX,
centroY, resolution, fov_horizontal, fov_vertical)

                posicao_original = ler_posicao_atual()

                cv2.putText(frame, "Processo iniciado...", (30, 80),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2, cv2.LINE_AA)
                cv2.putText(frame, f"Coordenadas: ({centroX}, {centroY}), ID:
{int(classes[i])}", (30, 110), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2,
cv2.LINE_AA)
```

```
                if classes[i] == 3:
                    # Ação para o ID 1
                    print("Ação para o ID 1")
                    enviar_comando_movimento_linear(-0.18865679639057054, -
0.2774107649261664, 0.13147904240505365, -3.092836056635589,
0.18254090607574172, -0.015895684687669248)

                    time.sleep(2)

                    posicao_inicial = ler_posicao_atual()
                    rx, ry, rz = posicao_inicial[3], posicao_inicial[4],
posicao_inicial[5]

                    enviar_comando_movimento_linear(posicao_inicial[0] +
dist_x_metros, posicao_inicial[1] - dist_y_metros, posicao_inicial[2], rx, ry,
rz)

                    time.sleep(2)
                    enviar_comando_movimento_linear(posicao_inicial[0] +
dist_x_metros, posicao_inicial[1] - dist_y_metros, posicao_inicial[2]-0.08,
rx, ry, rz)

                    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
s:
                        s.connect((HOST, PORT))
                        c = closegripper.read(1024)
                        time.sleep(1)
                        while c:
                            s.send(c)
                            print("close")
                            c = closegripper.read(1024)
                        time.sleep(3)

                    enviar_comando_movimento_linear(posicao_inicial[0] +
dist_x_metros, posicao_inicial[1] - dist_y_metros, posicao_inicial[2], rx, ry,
rz)

                    time.sleep(3)
                    enviar_comando_movimento_linear(0.3479285815924197, -
0.10862792553566582, 0.18226887966114105, 2.808015844827552, 1.12683285163102,
0.02193895727105998)
                    time.sleep(3)
                    enviar_comando_movimento_linear(0.3184600737291718, -
0.1123164360686253, 0.09581443228210906, 3.0544607705406905, -
0.21303549930530621, -0.06357220287730588)

                    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
s:
                        s.connect((HOST, PORT))
                        c = opengripper.read(1024)
                        while c:
                            s.send(c)
                            print("open")
                            c = opengripper.read(1024)
                        time.sleep(3)

                    enviar_comando_movimento_linear(0.3479285815924197, -
0.10862792553566582, 0.18226887966114105, 2.808015844827552, 1.12683285163102,
0.02193895727105998)
                    enviar_comando_movimento_linear(posicao_original[0],
posicao_original[1], posicao_original[2], posicao_original[3],
posicao_original[4], posicao_original[5])

                elif classes[i] == 1:
                    # Ação para o ID 2
                    print("Ação para o ID 2")
                    # Caso queira adicionar ação para múltiplos tipo
```

```
                # Adicione mais elif para outros IDs conforme necessário

                print("Comando enviado para mover o robô até o objeto
detectado.")

    desenhar_matriz(frame, 10, 10)
    cv2.putText(frame, 'FPS: {0:.2f}'.format(frame_rate_calc), (30, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2, cv2.LINE_AA)
    cv2.putText(frame, f'Objetos detectados: {num_objects_detected}', (30,
140), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2, cv2.LINE_AA)
    cv2.imshow('Object detector', frame)

    t2 = cv2.getTickCount()
    time1 = (t2 - t1) / freq
    frame_rate_calc = 1 / time1

    if key == ord('q'):
        break

cv2.destroyAllWindows()
videostream.stop()
```

A.2 Código para testar posição do UR3e

```
import socket
import struct
import time

HOST = "169.254.148.38"
PORT = 30003

def ler_posicao_atual(s):

    data = s.recv(1112)

    pos = struct.unpack('!6d', data[444:492])
    return pos

def main():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))

        time.sleep(1)
        posicao_atual = ler_posicao_atual(s)
        print("Posição atual do robô (juntas):", posicao_atual)


if __name__ == "__main__":
    main()
```

A.3 Código para tirar fotos com a PiCamera

```
from picamera2 import Picamera2
from threading import Thread
import cv2
import numpy as np
import os


class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self, resolution=(1280,720), framerate=30):
        self.picam2 = Picamera2()
        self.preview_config =
self.picam2.create_preview_configuration(main={"format": "XRGB8888", "size":
(1280,720)})
        self.picam2.configure(self.preview_config)
```

```
        self.frame = None
        self.stopped = False
        self.resolution = resolution

    def start(self):
        self.picam2.start()
        Thread(target=self.update, args=()).start()
        return self

    def update(self):
        while True:
            if self.stopped:
                return
            self.frame = self.picam2.capture_array()

    def read(self):
        return self.frame

    def stop(self):
        self.stopped = True
        self.picam2.stop()

# Função para salvar imagens
def save_image(frame, img_count):
    save_path = "/home/senha123/tflite1/calibrar_camera/fotos/teste" ##ajustar
local que deseja salvar as fotos
    if not os.path.exists(save_path):
        os.makedirs(save_path)
    filename = os.path.join(save_path, f"calibration_image_{img_count}.jpg")
    cv2.imwrite(filename, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    print(f"Foto {img_count} capturada e salva como {filename}")

# Inicializa e inicia o stream de vídeo
videostream = VideoStream(resolution=(640, 480), framerate=30).start()

img_count = 0
max_images = 50  # Número máximo de imagens para capturar

cv2.namedWindow("Frame")

while True:
    frame = videostream.read()
    if frame is not None:
        cv2.imshow("Frame", frame)

    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        break
    elif key == ord('p') and img_count < max_images:
        img_count += 1
        save_image(frame, img_count)
        if img_count == max_images:
            print("Número máximo de imagens capturadas.")
            break

videostream.stop()
cv2.destroyAllWindows()
```

A.4 Código para calibração

```
import cv2
import numpy as np
import glob
import os


pasta_imagens = "C://tflite1//CameraCalibration//fotos_capturadas"
```

```
padrao_tamanho = (8, 8)
tamanho_quadrado = 0.11


objp = np.zeros((padrao_tamanho[0] * padrao_tamanho[1], 3), np.float32)
objp[:, :2] = np.mgrid[0:padrao_tamanho[0], 0:padrao_tamanho[1]].T.reshape(-1,
2)
objp *= tamanho_quadrado


pontos_objeto = []
pontos_imagem = []


imagens = glob.glob(os.path.join(pasta_imagens, '*.png'))


if not imagens:
    print(f"Nenhuma imagem encontrada na pasta {pasta_imagens}. Verifique o
caminho e as imagens.")
else:
    print(f"{len(imagens)} imagens encontradas.")


for caminho_imagem in imagens:
    imagem = cv2.imread(caminho_imagem)
    if imagem is None:
        print(f"Falha ao carregar a imagem {caminho_imagem}.")
        continue

    cinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)


    ret, cantos = cv2.findChessboardCorners(cinza, padrao_tamanho, None)

    if ret:
        pontos_objeto.append(objp)
        pontos_imagem.append(cantos)


        cv2.drawChessboardCorners(imagem, padrao_tamanho, cantos, ret)
        cv2.imshow('Cantos do Tabuleiro de Xadrez', imagem)
        cv2.waitKey(500)
    else:
        print(f"Cantos do padrão de xadrez não encontrados na imagem
{caminho_imagem}.")

cv2.destroyAllWindows()


if len(pontos_objeto) == 0 or len(pontos_imagem) == 0:
    print("Nenhum canto de padrão de xadrez encontrado em nenhuma imagem.
Verifique as imagens e o padrão.")
else:

    ret, matriz_camera, coef_distorcao, rvecs, tvecs =
cv2.calibrateCamera(pontos_objeto, pontos_imagem, cinza.shape[::-1], None,
None)


    print("Matriz da Câmera:")
    print(matriz_camera)

    print("\nCoeficientes de Distorção:")
    print(coef_distorcao)
```

```
    np.savez('calibracao_camera.npz', matriz_camera=matriz_camera,
coef_distorcao=coef_distorcao)

    print("\nCalibração concluída. Resultados salvos em
'calibracao_camera.npz'.")
```

A.5 Criar um tabuleiro de xadrez para calibrar

```
import matplotlib.pyplot as plt
import numpy as np

def generate_chessboard(board_size=8, square_size=50):
    # Dimensões da imagem do tabuleiro
    img_size = board_size * square_size
    # Criação de uma imagem vazia
    chessboard = np.zeros((img_size, img_size), dtype=np.uint8)

    # Preenchimento do tabuleiro com quadrados brancos e pretos
    for i in range(board_size):
        for j in range(board_size):
            if (i + j) % 2 == 0:
                chessboard[i*square_size:(i+1)*square_size,
j*square_size:(j+1)*square_size] = 255

    return chessboard

def save_chessboard_image(filename="chessboard.png", board_size=8,
square_size=50):
    chessboard = generate_chessboard(board_size, square_size)
    plt.imshow(chessboard, cmap='gray', interpolation='nearest')
    plt.axis('off')  # Esconde os eixos
    plt.savefig(filename, bbox_inches='tight', pad_inches=0)
    print(f"Imagem do tabuleiro de xadrez salva como {filename}")

save_chessboard_image(board_size=9, square_size=50)
```