INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA MECÂNICA




VISÃO 3D NA AUTOMATIZAÇÃO DE PROCESSOS LOGÍSTICOS

MIGUEL DELGADO SILVEIRA
julho de 2024

POLITÉCNICO
DO PORTO

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA MECÂNICA




VISÃO 3D NA AUTOMATIZAÇÃO DE PROCESSOS LOGÍSTICOS

MIGUEL DELGADO SILVEIRA
Junho de 2024

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

VISÃO 3D NA AUTOMATIZAÇÃO DE PROCESSOS LOGÍSTICOS

Miguel Delgado Silveira

Dissertação para obtenção do Grau de Mestre em Engenharia Mecânica, Área de Especialização em Construções Mecânicas

Orientador: Adriano Manuel de Almeida Santos

Co-orientador: Filipe Alexandre De Sousa Pereira

Júri:

Presidente:

António José de Sousa Ferreira da Silva, Professor Adjunto, ISEP

Vogais:

António José Ramos Silva, Professor Auxiliar, FEUP

Adriano Manuel de Almeida Santos, Professor Adjunto, ISEP

Porto, junho 2024

Agradecimentos

Gostaria de expressar a minha profunda gratidão às pessoas que me apoiaram e contribuíram para a realização deste trabalho. Cada uma delas foi uma peça essencial, funcionando como componentes fundamentais, permitindo que eu, como um robô, executasse minhas tarefas com precisão e eficácia.

Em primeiro lugar, agradeço ao meu orientador, Prof. Adriano Santos, e ao meu coorientador, Prof. Filipe Pereira, que atuaram como o controlador central deste "robô". Com a sua orientação precisa e comandos claros, garantiram que cada movimento fosse calculado e eficaz, direcionando-me sempre pelo caminho certo.

À minha namorada, Kristina Đukić, que foi o que deu uma alma a este "robô". Apesar da distância, através da sua ajuda e constante companhia deram vida e propósito ao meu trabalho, transformando a simples execução de tarefas em algo com significado.

Por último agradeço aos meus pais, Graça Delgado e Luís Silveira, que funcionaram como a base sólida e estável deste "robô". A sua força e suporte inabaláveis proporcionaram a estabilidade e a energia necessárias para que eu pudesse realizar cada movimento com precisão e confiança. Eles foram o motor que alimentou toda a minha jornada, permitindo que eu alcançasse o ponto onde estou agora.

A todos, deixo aqui o meu mais sincero obrigado.

Resumo

Esta tese tem como objetivo explorar a aplicação de técnicas de visão 3D na automação de processos logísticos utilizando *deep learning* para a identificação e manipulação de objetos em ambientes industriais. O trabalho desenvolve um sistema de visão 3D que emprega modelos de deteção de objetos e *keypoints* treinados com ferramentas como *Roboflow* e *YOLOv8*. A metodologia inclui a recolha e anotação de dados, desenvolvimento de modelos de *deep learning* e análise dos resultados obtidos. Os modelos demonstraram uma precisão média (mAP) de 99.0%, uma precisão de 97.9% e um *recall* de 96.4% na identificação de blocos. Para a deteção de *keypoints*, o modelo alcançou um mAP de 98.3%, uma precisão de 96.4% e um *recall* de 95.6%. A integração dos modelos apresentou desafios computacionais, mas a abordagem combinada mostrou-se eficaz na identificação precisa de objetos. As limitações incluem a necessidade de otimização de recursos e aprimoramento dos processos de anotação. Futuras pesquisas devem focar-se na deteção da orientação dos blocos para aplicações em braços robóticos.

Palavras-chave: Visão 3D, Automação Logística, *Roboflow*, Deteção de Objetos, *Keypoints, YOLOv8*

Abstract

This thesis aims to explore the application of 3D vision techniques in the automation of logistical processes using deep learning for object identification and manipulation in industrial environments. The work develops a 3D vision system that employs object and keypoint detection models trained with tools such as Roboflow and YOLOv8. The methodology includes data collection and annotation, development of deep learning models, and analysis of the obtained results. The models demonstrated a mean Average Precision (mAP) of 99.0%, a precision of 97.9%, and a recall of 96.4% in block identification. For keypoint detection, the model achieved a mAP of 98.3%, a precision of 96.4%, and a recall of 95.6%. Integrating the models presented computational challenges, but the combined approach proved effective in precise detection. Limitations include the need for resource optimization and improvement in annotation processes. Future research should focus on detecting the orientation of blocks for applications in robotic arms.

KEYWORDS: 3D Vision, Logistics Automation, Roboflow, Object Detection, Keypoints, YOLOv8

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

ISEP	Instituto Superior de Engenharia do Porto
P.Porto	Instituto Politécnico do Porto
YOLO	*You Only Look Once*
ToF	*Time of Flight*
SCARA	*Selective Compliance Assembly Robot Arm*
DNN	*Deep Neural Network*
mAP	*Mean Average Precision*
IoU	*Intersection over Union*
FPS	*Frames Per Second*
IMU	*Inertial Measurement Unit*
6DoF	*Six Degrees of Freedom*

Lista de Símbolos

X, Y, Z	Coordenadas cartesianas	
θ	Ângulo de rotação	graus
d	Distância	m
F	Força	N
v	Velocidade	m/s
a	Aceleração	m/s^2
ρ	Densidade	kg/m^3
λ	Comprimento de onda	nm
Δ	Variação	

1. Introdução

Esta tese tem como objetivo explorar a aplicação da visão 3D na automação de processos logísticos, abordando especificamente a utilização de técnicas de *deep learning* para a identificação e manipulação de objetos em ambientes industriais. A seguir, será apresentada uma introdução detalhada que inclui a contextualização, os objetivos, a metodologia e a estrutura do relatório.

1.1. Contextualização

A automação de processos logísticos tem-se tornado uma área de crescente interesse devido à necessidade de aumentar a eficiência e reduzir os custos operacionais. Com o avanço da tecnologia, especialmente nas áreas de visão e inteligência artificial, é possível implementar sistemas mais inteligentes e adaptativos. A visão 3D, em particular, oferece um potencial significativo para melhorar a precisão e a flexibilidade de sistemas automatizados, permitindo a deteção e manipulação de objetos com maior eficácia. Este contexto tecnológico impulsiona a pesquisa e o desenvolvimento de soluções inovadoras que podem ser integradas nos processos industriais, proporcionando ganhos substanciais em produtividade e qualidade.

Um exemplo significativo é o "*Amazon Picking Challenge*", onde equipas desenvolveram soluções robóticas para tarefas de *pick and place* em armazéns. As equipas enfrentaram desafios como a manipulação de objetos de diferentes formas, tamanhos e materiais, e a integração de sistemas de visão artificial precisos e eficientes. Técnicas como o uso de algoritmos de visão artificial, incluindo YOLO, foram destacadas pela sua eficácia na deteção e reconhecimento de objetos.

Outro estudo relevante é o "*Pick and Place Robotic Arm: A Review Paper*", que oferece uma revisão abrangente sobre braços robóticos de *pick and place*. Este trabalho discute tecnologias de controle, sensores de *feedback* e a integração com sistemas de visão artificial. A combinação dessas tecnologias demonstrou melhorar a precisão, velocidade e adaptabilidade dos sistemas robóticos.

Apesar dos avanços significativos, ainda existem lacunas que precisam ser abordadas. Um dos principais desafios é a complexidade computacional envolvida na integração de múltiplos modelos de visão artificial. A maioria dos trabalhos anteriores foca-se na deteção de objetos ou na manipulação, mas poucos exploram a integração simultânea de modelos de deteção de objetos e *keypoints* para uma solução mais completa.

1.2. Objetivos

Os principais objetivos desta tese embora se possam considerar múltiplos tornaram-se muito específicos. Assim sendo, em primeiro lugar, pretendeu-se desenvolver um sistema de visão 3D capaz de identificar e manipular objetos num ambiente logístico, utilizando tecnologias de ponta em *deep learning*. Para alcançar este objetivo, foi implementado e treinado um conjunto de modelos de deteção de objetos e pontos-chave, utilizando ferramentas como *Roboflow* e YOLOv8. Além disso, foi fundamental avaliar a eficácia do sistema em diferentes cenários, considerando variáveis como iluminação, disposição dos objetos e a complexidade das tarefas. Outro objetivo importante foi propor melhorias e futuras aplicações da tecnologia desenvolvida no contexto industrial, identificando as áreas de maior impacto e potencial de inovação.

1.3. Metodologia

Para alcançar os objetivos propostos, foi adotada uma metodologia robusta e sistemática, que inclui várias etapas cruciais. A primeira etapa consiste na revisão bibliográfica, onde será realizada uma análise aprofundada das tecnologias e métodos atuais na automação logística e visão 3D. Esta revisão forneceu uma base teórica sólida para o desenvolvimento do trabalho. Em seguida, procedeu-se à coleta e anotação de dados, onde foram capturadas imagens de objetos logísticos e realizadas anotações manuais utilizando a plataforma Roboflow. Esta etapa foi essencial para a criação de um conjunto de dados de alta qualidade para o treino dos modelos.

A terceira etapa envolveu o desenvolvimento dos modelos de *deep learning*. Nesta fase, foram criados e treinados os modelos específicos para a deteção de objetos e pontos-chave, utilizando o YOLOv8. A configuração dos parâmetros de treino e a validação dos modelos foram realizadas para garantir a precisão e a robustez das previsões. Posteriormente, foram realizados testes de validação em condições reais, ajustando os parâmetros conforme necessário para otimizar o desempenho do sistema.

Finalmente, foi realizada uma análise dos resultados obtidos, discutindo-se o desempenho dos modelos e identificando-se áreas para melhorias futuras. Esta análise permitiu compreender os pontos fortes e fracos do sistema desenvolvido, fornecendo *insights* valiosos para a evolução da pesquisa.

1.4. Estrutura do Relatório

Este relatório está estruturado de forma a proporcionar uma compreensão clara e organizada dos passos seguidos durante a pesquisa, permitindo acompanhar o desenvolvimento do trabalho de maneira lógica e coesa. O relatório está dividido nos seguintes capítulos:

Introdução: Apresenta a contextualização, objetivos, metodologia e estrutura do trabalho. Este capítulo estabelece o enquadramento inicial e define as bases sobre as quais a pesquisa foi desenvolvida.

Revisão Bibliográfica: Explora o estado da arte em robótica industrial, sistemas colaborativos, Visão Artificial (VA) e técnicas de *deep learning*. Este capítulo fornece uma visão detalhada das tecnologias existentes e das suas aplicações no contexto da automação aplicada à logística.

Métodos e Aplicação: Detalha a simulação do processo logístico, as ferramentas e tecnologias utilizadas e o processo de desenvolvimento dos modelos de visão artificial. Serão apresentados os procedimentos adotados para a recolha de dados, anotação, treino e validação dos modelos.

Resultados e Discussão: Apresenta os resultados obtidos com os modelos desenvolvidos, discutindo o seu desempenho e eficácia. Neste capítulo analisou-se as métricas de desempenho, identificando-se os desafios enfrentados e as soluções implementadas para otimizar os resultados.

Conclusão: Resume as principais conclusões do trabalho, destacando as contribuições, limitações e sugestões para pesquisas futuras. Este capítulo encerrará o relatório, refletindo sobre os resultados alcançados e apontando caminhos para o desenvolvimento contínuo da pesquisa.

Esta estrutura visa proporcionar uma narrativa coerente e detalhada, que permitirá compreender todas as etapas do desenvolvimento da pesquisa, desde a conceção inicial até a análise dos resultados.

2. Revisão Bibliográfica

Neste capítulo, foi realizada uma revisão bibliográfica detalhada sobre os principais temas e tecnologias relacionados à aplicação de visão 3D na automação de processos logísticos. A revisão bibliográfica tem como objetivo fornecer um panorama abrangente e atualizado das técnicas e metodologias que fundamentam a pesquisa desenvolvida nesta tese. Através da análise crítica da literatura existente, pretende-se identificar os avanços, desafios e lacunas que ainda precisam ser abordados na área.

2.1. Robótica Industrial

Este ponto começa com uma breve introdução à Robótica Industrial, explorando os elementos fundamentais que a caracterizam e delineando a sua relevância na automação dos processos logísticos. Neste segmento, foi apresentada uma definição abrangente de Robótica Industrial, destacando-se o seu papel na otimização de tarefas industriais e sua evolução ao longo do tempo. Além disso, foi realizada uma análise detalhada com o intuito de identificar e categorizar os diversos tipos de robôs industriais, incluindo robôs articulados, cartesianos, SCARA, e outros tipos específicos. Esta seção pretende proporcionar uma compreensão clara da variedade de robôs utilizados na indústria. Exploramos também as inúmeras aplicações práticas da Robótica Industrial, abrangendo o papel fundamental desses sistemas em linhas de montagem automatizadas, processos de soldadura e manipulação de materiais, pintura industrial, embalagem e paletização. Detalharemos ainda os sensores e atuadores específicos utilizados na Robótica Industrial, incluindo sensores de visão, sensores táteis, e a utilização de atuadores hidráulicos e pneumáticos. A compreensão desses componentes é essencial para a eficiência operacional e segurança dos sistemas, fornecendo uma base sólida para a análise e discussão abordadas no contexto deste projeto.

2.1.1. Definição

Um robô Industrial é um dispositivo mecânico que pode ser programado para executar uma variedade de tarefas de manipulação e locomoção sob controle automático. O atributo "industrial" distingue esses robôs de produção e serviço dos robôs de ficção científica e dos robôs puramente de software (como os motores de busca da World Wide Web) [1].

Os robôs industriais são projetados para realizar operações de maneira rápida, repetitiva e precisa, desempenhando um papel significativo na produção industrial [2].

Um robô industrial é composto por várias partes. O manipulador mecânico, focado principalmente no aspeto mecânico e estrutural do robô, consiste na combinação de elementos estruturais rígidos, como corpos ou elos, conectados por articulações (juntas). Os atuadores desempenham a função de converter energia elétrica, hidráulica ou pneumática em potência mecânica, sendo essa potência transmitida aos elos por sistemas de transmissão para possibilitar seus movimentos. Os sensores fornecem parâmetros sobre o comportamento do manipulador, incluindo posição e velocidade dos elos ao longo do tempo, assim como informações sobre a interação do robô com o ambiente operativo, como força, binário e sistemas de visão. A unidade de controle é responsável pela gestão e supervisão dos parâmetros operacionais necessários para realizar as tarefas do robô. A unidade de potência é responsável por fornecer a potência necessária para a movimentação dos atuadores, por sua vez, atua como o elemento de ligação entre o robô e o ambiente circundante, podendo ser uma garra ou ferramenta [3].

2.1.2. Tipos de robôs Industriais

No âmbito da robótica industrial, a distinção entre tipos de robôs encontra-se intrinsecamente ligada à sua estrutura cinemática, constituindo um fator determinante na sua capacidade de desempenhar tarefas específicas. Essencialmente, existem cinco categorias principais de robôs industriais, cada uma com características distintas e aplicações especializadas. Estas categorias incluem:

- Robôs cartesianos, que operam em coordenadas lineares;

- Robôs cilíndricos, com movimentos rotativos e lineares;

- Robôs esféricos, que se destacam pela sua capacidade de movimento esférico;

- Robôs articulados horizontais, conhecidos como SCARA, reconhecidos pela sua precisão em movimentos horizontais;

- Robôs articulados verticais antropomórficos, que replicam a articulação humana e são versáteis em diversas tarefas [4, 5].

Um robô cartesiano é um robô industrial com três eixos principais de controle - X, Y e Z – que são perpendiculares entre si. Os eixos X e Y formam uma superfície planar paralela ao solo, enquanto o eixo Z é responsável pelo movimento vertical. Os atuadores montados em cada eixo são responsáveis por acionar os eixos correspondentes. Tipicamente, um motor elétrico combinado com outros dispositivos mecânicos, como caixas de engrenagens, acoplamentos, polias, correias, etc., realiza o movimento de translação de um eixo. Alguns sistemas de pórtico exigem dois motores montados no eixo de base, pois solicitam mais binário para superar o peso do sistema. Essa configuração geralmente exige mais dos programadores, pois ambos os motores elétricos devem ser sincronizados, ou seja, operam com os mesmos valores cinemáticos em um intervalo de tempo sincronizado, que inclui distância, velocidade, aceleração e desaceleração [6].

Os Robôs Lineares Cartesianos, com três eixos lineares, oferecem uma estrutura estável e de fácil compreensão, tornando-os ideais para aplicações no processamento de chapas metálicas.

A sua rigidez, programação *offline* direta e precisos stops mecânicos aprimoram sua utilidade em tarefas que requerem precisão. Apesar de suas vantagens, os Robôs Lineares Cartesianos apresentam limitações, incluindo um alcance restrito para a frente e para trás, a necessidade de um espaço significativo, desafios na vedação eficaz e um custo relativamente mais elevado [7].

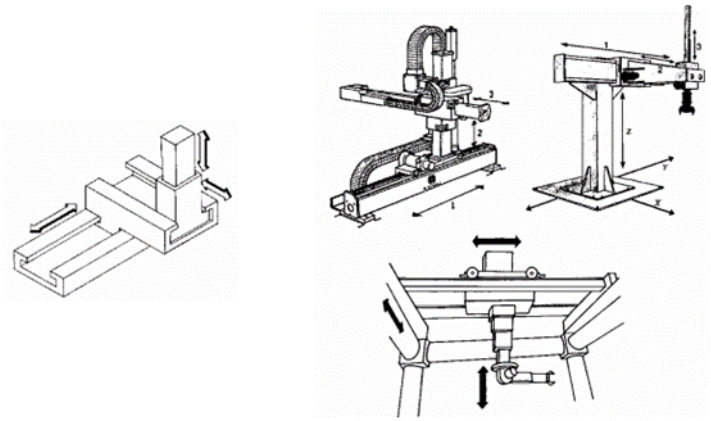


Figura 1- Manipulador Cartesiano [5]

Os robôs cilíndricos são caracterizados por duas juntas prismáticas e uma junta rotativa, formando uma configuração ideal para tarefas de posicionamento. O manipulador final do robô cria um espaço cilíndrico de trabalho, conseguido através da montagem de um braço horizontal capaz de movimentos para frente e para trás. Este braço horizontal está ligado a um carro que se move verticalmente, conectado à base rotativa. O esquema do robô e sua representação são ilustrados na Figura *2*, e devido ao movimento de ambas as unidades na base, o espaço de trabalho do robô forma um espaço anelar ao redor do cilindro. Esses robôs têm amplo uso no fabrico de sistemas eletrônicos [7].

Os Robôs Cilíndricos, com um eixo de rotação e dois eixos lineares, oferecem a capacidade de alcance completo ao redor. Suas características incluem dois eixos rígidos, facilidade de vedação num dos eixos e movimento horizontal circular. Apesar das vantagens, os Robôs Cilíndricos apresentam limitações, como a incapacidade de alcançar acima de si mesmos, um eixo menos rígido, dificuldades de vedação em dois dos eixos, e a impossibilidade de contornar obstáculos. Amplamente utilizados em operações de montagem, manipulação em máquinas-ferramenta, soldadura por pontos e manipulação de máquinas de fundição. A adequação desses robôs varia conforme as necessidades específicas de cada aplicação industrial [7].

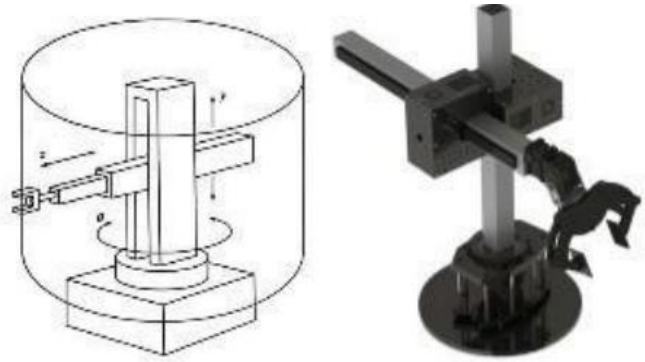


Figura 2- Manipulador Cilíndrico [7]

Robôs esféricos, conhecidos informalmente como robôs polares, destacam-se pelo porte considerável e pela presença de um braço telescópico. Os movimentos fundamentais de um robô esférico englobam rotação na base e articulação angular vertical ao longo do braço. Com um mínimo de duas juntas móveis e uma junta fixa, esses robôs exibem uma estrutura robusta. A Figura *3* elucida o diagrama esquemático e a representação simbólica de um robô esférico. A cinemática do robô abrange uma sequência de movimento tripartida: o primeiro movimento envolve rotação da base ao longo do eixo vertical; o movimento subsequente implica ajustes rotacionais do braço; e, finalmente, o terceiro movimento orquestra o movimento de vai e vem do braço [7].

Os Robôs Esféricos, com dois eixos de rotação e um eixo linear, oferecem a capacidade de alcance completo ao redor e acima ou abaixo de obstáculos. Possuem um grande volume de trabalho e são frequentemente utilizados na indústria eletrônica para pegar e posicionar objetos. Apesar das vantagens, os Robôs Esféricos apresentam limitações, como a incapacidade de alcançar acima de si mesmos, alcance vertical reduzido e curto. Amplamente utilizados na indústria eletrônica para tarefas de pegar e posicionar objetos [7].

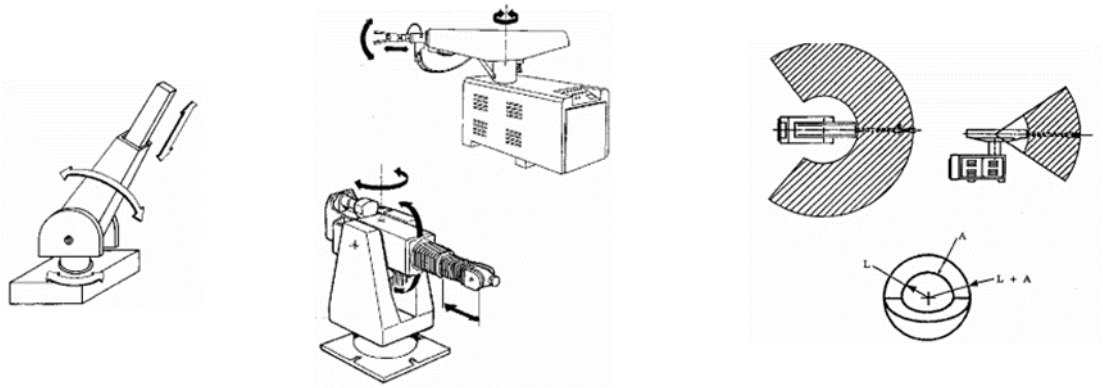


Figura 3- Manipulador Cilíndrico [5]

Um robô articulado horizontal ou SCARA (Selective Compliance Assembly Robot Arm) é composto por duas juntas rotativas paralelas e uma junta prismática, permitindo movimentos ao longo dos planos horizontal e vertical. O SCARA apresenta operação suave ao longo dos eixos X e Y, enquanto exibe significativa resistência ao longo do eixo Z. Os robôs SCARA são reconhecidos por sua capacidade de operar suavemente ao longo dos eixos X e Y, ao mesmo

tempo em que proporcionam resistência substancial ao longo do eixo Z. O braço SCARA é capaz de pegar verticalmente uma peça de uma mesa posicionada horizontalmente, mover-se horizontalmente para um ponto específico e realizar tarefas de montagem ao baixar o braço e posicionar a peça em seu local adequado. A Figura *4* representa um típico robô SCARA [7].

Os Robôs SCARA, com dois eixos de rotação e um eixo linear, oferecem alcance completo ao redor e acima ou abaixo de obstáculos, além de possuir um grande volume de trabalho. São comumente utilizados para apertar parafusos, transferir peças pesadas e realizar inspeções. Apesar das vantagens, os Robôs SCARA têm limitações, como a incapacidade de alcançar acima de si mesmos e um alcance vertical reduzido. Estes robôs são amplamente utilizados para apertar parafusos, transferir peças pesadas e realizar inspeções [7].

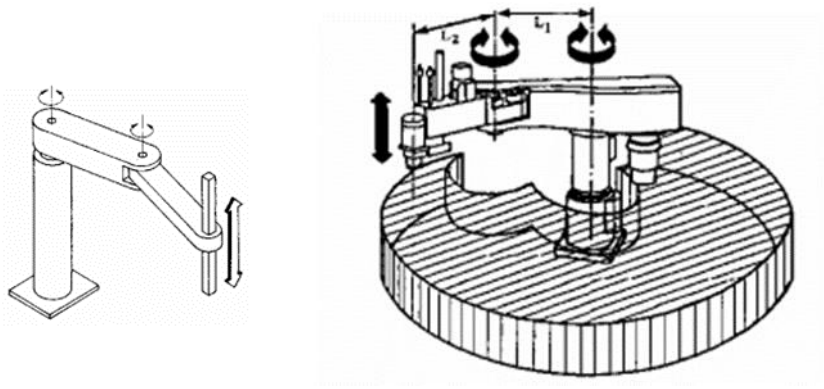


Figura 4- Manipulador Articulado Horizontal [5]

Robôs articulados, comumente conhecidos como robôs revolutos, destacam-se por seus três eixos fixos conectados a duas bases revolutas, espelhando a biomecânica do braço humano. As juntas do braço articulado são revolutas, assemelhando-se a dobradiças, e os componentes móveis são denominados elos. A Figura *5* apresenta de forma elucidativa o diagrama esquemático e representação de um robô articulado. As juntas revolutas replicam dobradiças, enquanto as juntas prismáticas desempenham o papel de componentes deslizantes. Cada junta orquestra o movimento relativo de objetos interconectados, delineando um subconjunto no espaço de configuração. A dimensão do espaço de configuração expande-se com o aumento do número de juntas, impactando a velocidade operacional devido a cargas variáveis e fatores ambientais não lineares [7].

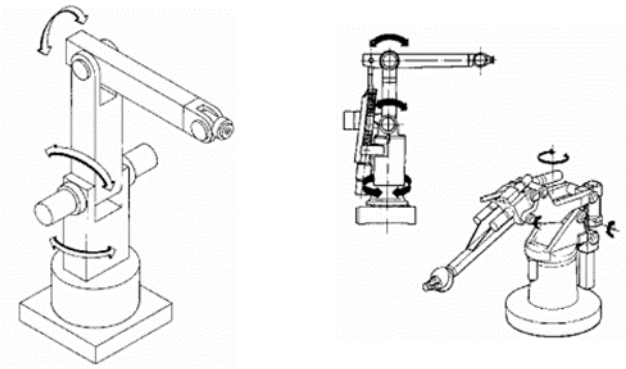


Figura 5- Manipulador Articulado Vertical [5]

2.1.3. Aplicações na Indústria

O próximo segmento de nossa análise dedica-se às diversas aplicações da robótica na indústria, destacando três pilares fundamentais: transporte, manipulação e sensoriamento. Abordaremos como os robôs desempenham papéis cruciais no eficiente deslocamento de materiais, na habilidade de manipulação para tarefas como paletização, e na integração de tecnologias sensoriais para aprimorar a perceção e precisão em ambientes industriais.

No âmbito do sistema de manufatura, uma das operações elementares realizadas nas peças durante sua trajetória é o transporte ou deslocamento físico. As peças são conduzidas de um ponto a outro, armazenadas, trabalhadas, montadas ou embaladas. No decorrer dessas operações de transporte, as características físicas da peça permanecem inalteradas. O robô emerge como um candidato ideal para tais operações. Tarefas simples de manuseamento de materiais, como a transferência de peças de um transportador para outro, podem requerer apenas movimentos unidimensionais ou bidimensionais. Esses tipos de operações são frequentemente desempenhados por robôs não servo. Outras atividades de manipulação de peças podem revelar-se mais intrincadas, exigindo maior destreza manipulativa e capacidade adicional de processamento, além da aptidão para o transporte [8]. Exemplos dessas tarefas mais complexas englobam o carregamento e descarregamento de máquinas, paletização, classificação de peças e embalagem. Essas operações são normalmente realizadas por robôs servo controlados ponto a ponto [1], Figura *6*.

Figura 6- Robô de transporte de carga [9]

Para além da manipulação de materiais, uma operação crucial no processo de transformação de uma peça, desde a sua condição de matéria-prima até um produto acabado, é o processamento, frequentemente requerendo algum tipo de manipulação. Por outras palavras, as peças são inseridas, orientadas ou ajustadas para se encontrarem na posição adequada para processos como maquinagem, montagem ou outras operações relevantes. Em muitos casos, é a ferramenta que é habilmente manipulada, em detrimento da própria peça em processo.

Exemplos ilustrativos compreendem a maquinagem assistida por robôs, soldadura como é possível ver na Figura *7* e a arco, tratamento térmico, união ou corte a laser, bem como a pintura por pulverização. Operações mais complexas, como a montagem, igualmente contam com a contribuição essencial do robô [1].

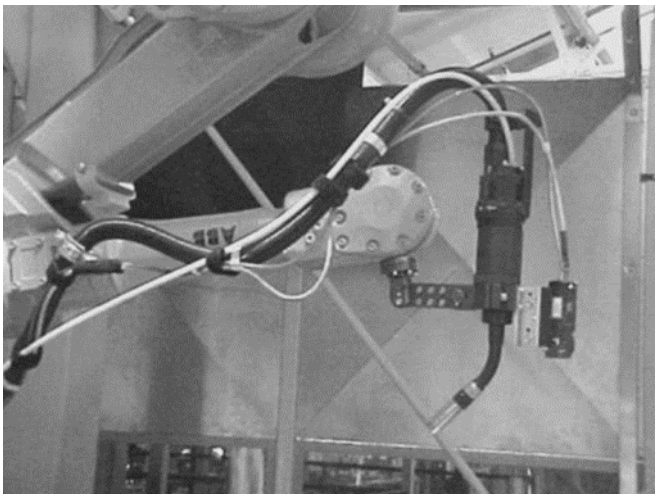


Figura 7- ABB 2400–16 Robô com Tocha de Soldagem MIG e o Sensor de Soldagem OST CSS montado na extremidade do braço (HKS Prozesstechnik 2013) [10]

Além do transporte e manipulação, a utilização de meios de feedback sensorial torna-se crucial em aplicações de processamento sofisticadas. Enquanto uma operação de soldadura por ponto pode prescindir de feedback, um processo de soldadura a arco requer um meio de rastreamento da costura. Esses inputs sensoriais podem provir de uma variedade de tipos de sensores, incluindo sensores de proximidade, sensores de força e sistemas de visão a laser [1].

Robôs na Manipulação de Materiais e Armazenamento

Manipulação de materiais consiste na aplicação cuidadosa do método apropriado para fornecer a quantidade precisa do material correto, na orientação adequada e nas condições apropriadas, ao local específico, no momento oportuno e com o custo adequado, preservando um ambiente de trabalho seguro. A definição precisa é de importância crucial para o êxito das aplicações bem-sucedidas de robôs na manipulação de materiais [1].

Os robôs de *pick-and-place* são amplamente utilizados em várias atividades de manipulação de materiais, abarcando desde a paletização e a seleção de caixas até o carregamento e descarregamento de máquinas, assim como outras funções, incluindo alimentação e entrega de peças [8]. Esses robôs também são usados em sistemas de armazenamento e recuperação, bem como em processos de empacotamento e ordenação de caixas [1].

Figura 8- Robô pick-and-place [1]

2.2. Sistema Colaborativo

2.2.1. Definição

A distinção crucial entre robôs colaborativos e os tradicionais robôs industriais reside na interação direta com os operadores humanos. A utilização dessa interação oferece às organizações a oportunidade teórica de combinar as capacidades dos robôs, como força e resistência, com o conhecimento tácito e as habilidades ágeis de tomada de decisão dos seres humanos. Desse modo, as organizações podem usufruir das vantagens cruciais apresentadas tanto pelos humanos quanto pelos robôs. Enquanto os robôs se destacam em tarefas repetitivas e monótonas, os operadores humanos ainda são mais eficazes em lidar com tarefas inesperadas e não planeadas. Em certo sentido, os humanos permanecem como o recurso mais flexível no sistema. Aproveitando essas vantagens heterogêneas, a colaboração entre humanos e robôs pode superar os processos puramente robóticos, como evidenciado em configurações de pesquisa experimental [11] [12].

2.2.2. Tipos de Sistemas Colaborativos

Existem quatro cenários distintos de colaboração [13]:

Independente: Neste cenário, um operador humano e um cobot (robô colaborativo) realizam trabalhos separados em peças distintas, de maneira independente, compartilhando o mesmo espaço de trabalho sem a necessidade de gaiolas ou cercas [14].

Simultâneo: Aqui, um operador humano e um cobot conduzem processos de fabricação distintos no mesmo objeto de trabalho ao mesmo tempo. Essa operação simultânea minimiza

o tempo de trânsito, melhora a produtividade e a utilização do espaço, sem depender temporalmente um do outro [14].

Sequencial: No cenário sequencial, um operador humano e um cobot executam processos de fabricação em sequência no mesmo objeto de trabalho. Existe uma dependência temporal entre as tarefas do operador e do cobot, com o cobot frequentemente designado para lidar com processos mais monótonos, melhorando assim as condições de trabalho do operador [14].

De apoio: Nesse caso, um operador humano e um cobot colaboram interactivamente no mesmo processo, no mesmo objeto de trabalho. Aqui, há uma dependência completa entre o humano e o cobot, pois um não pode realizar a tarefa sem o outro [14].

Atualmente, a maioria dos exemplos de implementação de cobots em ambientes industriais concentra-se nos cenários de colaboração "independente" ou "simultânea". Projetos de pesquisa mais avançados, que procuram inovação, muitas vezes procuram cenários de colaboração "sequencia" ou "de apoio" [13].

Figura 9- Exemplos de robôs colaborativos

2.2.3. Aplicações Práticas em Ambientes Industriais

Atualmente, os robôs colaborativos são utilizados em diversos setores, abrangendo desde intervenções médicas, onde os cobots desempenham funções cruciais em cirurgias complexas com elevada precisão, até a reabilitação, onde são empregues como sistemas terapêuticos, auxiliando a população com deficiência a realizar uma variedade de movimentos musculares e articulares. Na área de pesquisa e educação, diferentes cobots podem ser utilizados para repetir ações específicas, contribuindo para a verificação de sistemas particulares. Destaca-se, ainda, a relevância significativa dos robôs colaborativos na indústria, especialmente nas indústrias automóveis e linhas de montagem, onde desempenham diversas tarefas, desde a recolha, embalagem e paletização, até soldagem, montagem de itens, manipulação de materiais e inspeção de produtos, entre outras [15].

As aplicações dos cobots abrangem diversas áreas, destacando-se especialmente na seleção, empacotamento e palatização de itens, assim como na execução de tarefas de soldadura, sobretudo na indústria automóvel. Esses robôs colaborativos, capazes de operar com alta precisão e velocidade de maneira autônoma, desempenham um papel significativo ao facilitar as tarefas dos seres humanos em atividades como montagem de itens. Além disso, contribuem

para criar ambientes de trabalho mais seguros, manipulando materiais perigosos e realizando inspeções de produtos com precisão, sem os efeitos do cansaço humano. De forma abrangente, a implementação desses robôs resulta em benefícios notáveis, promovendo a saúde e segurança dos trabalhadores, reduzindo custos operacionais e otimizando os ciclos de produção [16], minimizando períodos de inatividade [17].

2.2.4. Normas e Regulamentações

"Operação colaborativa" refere-se a um estado no qual um robô projetado intencionalmente trabalha diretamente ao lado de uma ou mais pessoas dentro de um espaço de trabalho compartilhado, permitindo que eles realizem tarefas simultaneamente. Durante essa operação, a interação física entre o robô e um trabalhador é permitida. Por exemplo, a Figura *10* ilustra um exemplo de operação de alimentação de peças guiada manualmente. O operador guia o braço do robô até a posição da peça, guiando o braço para agarrar a peça dentro do espaço de trabalho colaborativo usando um dispositivo de orientação manual. Posteriormente, o operador move o braço para o espaço de operação automática. Uma vez que o braço ultrapassa a fronteira estabelecida por salvaguardas, o robô faz a transição para o modo de operação automática para executar um processo programado. Além disso, espera-se que as operações colaborativas incluam assistência em tarefas como manipulação de peças ou ferramentas em processos de montagem de peças ou soldadura.

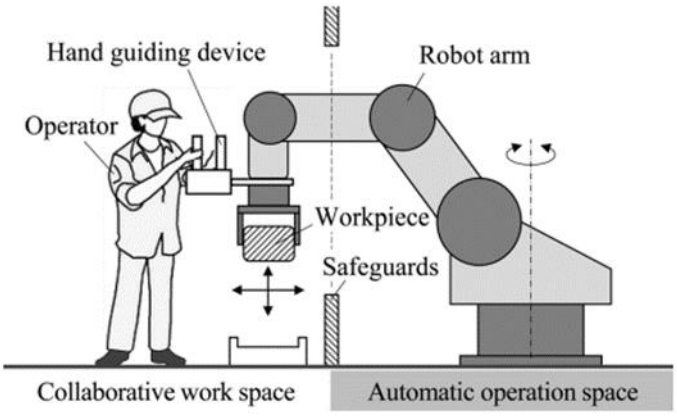


Figura 10- Exemplo de operação com robô colaborativo [18]

A *Tabela 1* para o funcionamento em colaboração apresenta medidas de redução de riscos para operações colaborativas listadas nas normas ISO 10218-1 e 2, que são alcançadas por funções relacionadas à segurança ou pelo design inerentemente seguro do robô. Essas normas exigem que a segurança dos trabalhadores seja garantida utilizando uma ou mais dessas medidas; no entanto, os conceitos ou esboços das medidas são apenas estipulados, conforme mostrado na *Tabela 1* para o funcionamento em colaboração [16]. Essas normas também afirmam que as especificações detalhadas das medidas, como limite de velocidade, precisão de paragem e tempo de resposta, devem ser determinadas adequadamente pelo projetista ou pelo integrador do sistema com base em suas avaliações de risco noutras normas ISO relevantes.

Isso será mencionado novamente mais tarde no contexto dos problemas ao introduzir os requisitos das normas ISO no OISH (Occupational Safety and Health Management System) [19].

Tabela 1- Medidas de redução dos riscos enumeradas na ISO 10218-1 e 2 para o funcionamento de robôs colaborativos [19]

Medidas	Descrição e requerimentos
Paragem monitorizada para avaliar segurança	Quando uma pessoa entra no espaço de trabalho colaborativo, o robô é obrigado a parar e manter a sua posição. Esta condição de imobilidade deve ser monitorizada por esta função de segurança, e se for detetada uma divergência superior a uma quantidade definida, a energia para acionar os atuadores é removida.
Controlo de velocidade	Esta função de segurança monitoriza a velocidade de movimento do robot. Se a velocidade exceder o limite predefinido, esta função para todos os movimentos do robô.
Limitação da potência	A potência ou força do robô é limitada por uma conceção intrinsecamente segura ou monitorizada por uma função de segurança nominal durante a operação de colaboração.
Dispositivo de guiamento manual	Quando aplicado, deve ser instalado um dispositivo de ativação e um dispositivo de paragem de emergência e a velocidade de movimento do robô deve ser monitorizada.
Controlo da distância de separação	Esta função de segurança controla se a distância de segurança entre o operador e o robot é mantida acima do valor predefinido.
Deteção de intrusões	A intrusão de qualquer pessoa, que não os operadores especificados (indivíduos treinados e autorizados a interagir com o robô dentro de seu espaço operacional), no espaço móvel do robot deve ser detetada, e isto deve provocar a paragem de todos os movimentos perigosos do robô.
Fornecimento de uma autorização adequada	O sistema de robot deve ser instalado de modo que haja um espaço livre de 500 mm ou mais entre o seu espaço móvel e o equipamento periférico, outras máquinas e/ou obstáculos. Se esta distância não puder ser assegurada, devem ser tomadas medidas de proteção adicionais.
Equipamento de proteção individual	Se necessário, devem ser tidas em conta as eventuais limitações do operador decorrentes da utilização do equipamento de proteção.

2.3. Visão Artificial

A perceção tridimensional representa uma das tecnologias fundamentais para robôs. Uma visão em três dimensões do ambiente circundante é crucial para a execução autônoma de tarefas de navegação e manipulação em ambientes parcialmente conhecidos. Além disso, a tele operação de robôs necessita de uma representação visual do ambiente de forma compreensível para os humanos, sendo essencial para uma interface intuitiva. Dessa forma, os sistemas de visão destinados à orientação de robôs necessitam, em geral, adquirir informações tridimensionais [20].

A projeção de um ponto da cena na imagem pode ser calculada matematicamente, como demonstrado na Figura *11*. No entanto, ao considerar um ponto específico na imagem, não é possível obter diretamente a sua correspondência única no espaço, pois essa relação não é de um para um, mas sim de um para vários (Figura *11*a). Portanto, o problema inverso torna-se mal definido em termos algébricos. A projeção de um ponto tridimensional na imagem não é uma aplicação injetora, permitindo que diferentes pontos sejam projetados no mesmo pixel. A solução do problema inverso resulta, na realidade, em uma linha reta formada por todos os pontos representados no mesmo pixel da imagem, como ilustrado na Figura 11b [21].

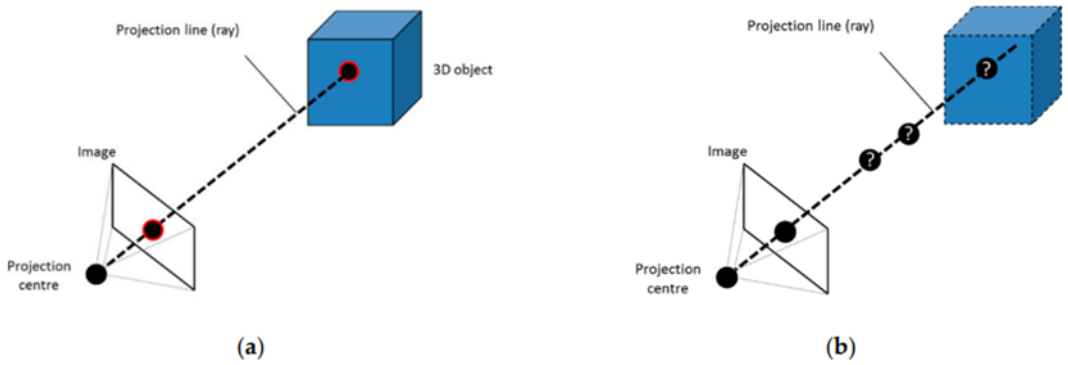


Figura 11- De 3D para 2D. (a) Problema direto; (b) Problema inverso [20]

O problema da perceção tridimensional é abordado por técnicas passivas, como visão estéreo ou fotogrametria, que utilizam iluminação ambiente, e por técnicas ativas, que projetam padrões visíveis ou infravermelhos e estimam informações de profundidade com base no tempo de retorno, deformação do padrão ou cálculos trigonométricos. Os sistemas de visão ativa têm a vantagem de fornecer iluminação controlada, mas podem apresentar imprecisões, especialmente em bordas ou superfícies variadas. Sistemas passivos exigem múltiplas câmaras para reconstrução 3D, enquanto os ativos podem usar uma única câmara. Ambas as abordagens têm sensibilidade a fatores externos, sendo a escolha entre elas dependente da aplicação específica e do custo. A classificação dessas técnicas varia na literatura, e a Tabela *2* destaca diferentes grupos, indicando as sobreposições entre eles [22].

Tabela 2- Classificação das técnicas de visão [20]

	Câmara Única	**Múltiplas Câmaras**
Visão Passiva	2D	Visão Estéreo e Fotogrametria
Visão Ativa	*Time of Flight* Luz estruturada Codificação da luz Triangulação laser	Luz estruturada Visão estéreo de textura projetada

2.3.1. Visão estéreo e fotogrametria

A fotogrametria, em sua essência, refere-se à medição de dimensões reais a partir de uma fotografia de um objeto. Trata-se de uma técnica de reconstrução tridimensional fundamentada em imagens convencionais em duas dimensões, comumente usada em áreas como arquitetura, topografia, geologia, arqueologia, engenharia e indústria. Tanto na visão estereoscópica quanto nas técnicas fotogramétricas, é imperativo identificar o mesmo ponto em outras imagens para calcular a interseção das linhas de projeção e, consequentemente, determinar a posição tridimensional (Figura *12*b). É recomendável que cada ponto seja identificado em pelo menos três imagens, visando assegurar a deteção e aprimorar a precisão. Destaca-se a importância de selecionar pontos homólogos (Figura *12a*) para obter a posição tridimensional correta [20].

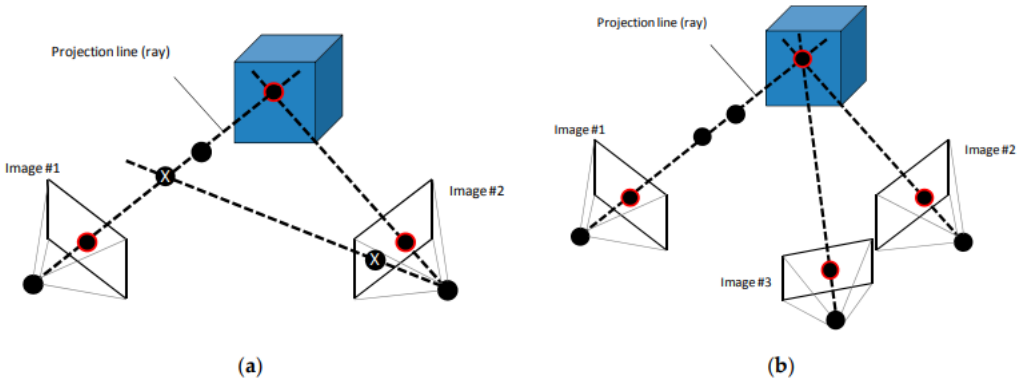


Figura 12- De 2D para 3D. (a) Pontos homólogos; (b) Intersecção das linhas de projeção [20]

2.3.2. Time of Flight (ToF)

As técnicas de visão ativa utilizam métodos como a projeção de padrões visíveis ou infravermelhos sobre objetos, conforme ilustrado na Figura *13*. O método Time of Flight (ToF), por exemplo, opera emitindo pulsos curtos de luz. Esses pulsos iluminam a cena, refletindo nos objetos. A câmara captura a luz refletida no plano do sensor. A largura do pulso da iluminação define a faixa máxima da câmara, tornando a unidade de iluminação um componente crucial. LEDs ou lasers especiais são frequentemente necessários para gerar pulsos suficientemente curtos para medições precisas.

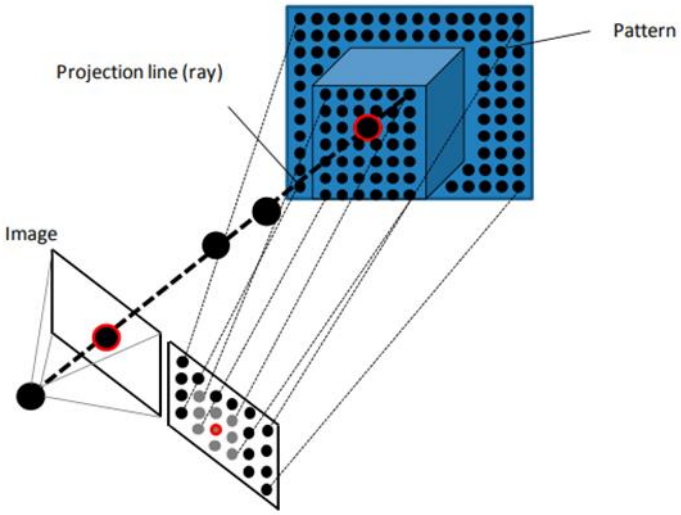


Figura 13- Projeção de um padrão num objeto [20]

2.3.3. Sensor de Luz Estruturada

O equipamento de luz estruturada é composto por uma fonte de luz (projetor de luz) e um ou dois recetores de informação (câmaras). Existem dois grupos principais de técnicas de luz estruturada: *time-multiplexing*, que envolve a projeção de uma sequência de padrões, e *one-shot*, que projeta um único padrão. O *time-multiplexing* permite alcançar uma alta resolução, mas requer que o objeto, o projetor e a câmara permaneçam estáticos. Por outro lado, as técnicas *one-shot* permitem o movimento, pois cada ponto ou linha codificado é identificado de forma única por meio de uma vizinhança local [23].

Geralmente, a luz projetada é luz branca, facilmente gerada e não é perigosa para as pessoas, ao contrário do laser. Essa luz é modificada por grades para criar linhas ou faixas com luzes e sombras, semelhantes a uma zebra Figura *14*, que são registadas pela câmara. A profundidade é obtida a partir das variações usando uma técnica semelhante à triangulação, que consiste em calcular a interseção entre planos e linhas [24].

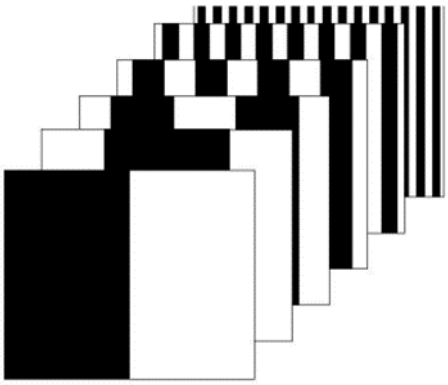


Figura 14- Padrões típicos de luz estruturada [20]

2.3.4. Codificação Luminosa

A codificação de luz utiliza uma abordagem totalmente diferente, onde a fonte de luz está constantemente ligada, reduzindo significativamente a necessidade de sincronização precisa das medições. Pode ser considerada uma evolução da luz estruturada. Uma fonte de laser emite luz invisível (aproximadamente na faixa de infravermelho) que passa por um filtro e é dispersa em um padrão semi-aleatório, mas constante, de pequenos pontos. O padrão refletido é então detetado por uma câmara de infravermelho e analisado. Com base no conhecimento do padrão de luz emitida, distorção da lente e distância entre emissor e recetor, a distância até cada ponto pode ser estimada medindo as deformações na forma e no tamanho dos pontos projetados [25].

A codificação de luz oferece dados de profundidade a um custo significativamente baixo, o que é uma grande inovação não apenas para a robótica. No entanto, ela possui algumas limitações, pois essas câmaras não fornecem um mapa de profundidade denso. As imagens de profundidade fornecidas contêm lacunas correspondentes às zonas onde o sensor enfrenta problemas, seja devido ao material dos objetos (reflexão, transparência, absorção de luz, etc.) ou à sua posição (fora de alcance, com oclusões, etc.). O mapa de profundidade é válido apenas para objetos que estão na faixa de 1 a 3 metros, a fim de reduzir o efeito de ruído e baixa resolução [26]. Além disso, como é baseado em um projetor de infravermelho com uma câmara de infravermelho, e como o sol emite no espectro infravermelho, a luz solar afeta negativamente o sistema [20].

2.3.5. Triangulação a Laser

Na triangulação a laser, o ponto, a câmara e o emissor de laser formam um triângulo Figura *15*. A distância entre a câmara e o emissor de laser é conhecida, e devido ao ângulo do canto do emissor de laser também ser conhecido, o ângulo do canto da câmara pode ser determinado ao observar a localização do ponto laser no campo de visão da câmara. Essas três informações determinam a forma e o tamanho do triângulo e fornecem a localização do canto do ponto laser do triângulo, que é, na verdade, o ponto 3D. A precisão depende da resolução do sensor CCD (Charge Coupled Device), da qualidade das lentes, do tamanho do ponto, da qualidade do feixe de laser, do estado da superfície da peça e de outros fatores óticos [27]. Uma desvantagem, dependendo da potência do laser, é que os sensores a laser podem ser perigosos para as pessoas. Eles não são seguros para os olhos [20].

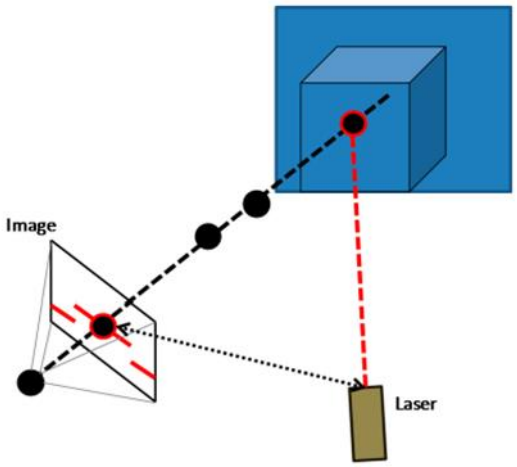


Figura 15- Triangulação a laser [20]

2.3.6. Comparação de Técnicas de Visão 3D

Dependendo do objetivo final da aplicação e do tipo de robô, diferentes considerações e fatores precisam ser levados em conta para selecionar a técnica de visão mais adequada:

•Precisão do alinhamento e resolução da nuvem de pontos. Esses são principalmente determinados pelo hardware (sensor) e software (extração, registo, segmentação, comparação, etc.), e estão em conformidade com o tamanho do objeto e o propósito da aplicação.

•Alcance do sensor. A distância de trabalho será determinada pela acessibilidade do robô, tamanho do sensor e configurações do ambiente.

•Peso leve. Se o sensor estiver a bordo ou montado no efetuador final, o robô possui uma carga máxima limitada para garantir sua dinâmica total.

•Questões de segurança. O robô pode trabalhar em proximidade com trabalhadores humanos, portanto, os sensores devem evitar lasers de alta potência perigosos para minimizar qualquer risco de acidentes.

•Tempo de processamento. O tempo de processamento pode ser crucial para determinar se um sistema é adequado para determinada aplicação, especialmente no caso de robôs em movimento com restrições de segurança, ou seja, a capacidade de detetar e evitar colisões com humanos ou obstáculos. Algumas técnicas requerem que o objeto e a câmara permaneçam estáticos para a captura, portanto, não são aplicáveis em cenários em movimento.

•Ambiente de digitalização. Condições de iluminação, vibrações, movimentos da câmara, etc., podem prejudicar a qualidade da nuvem de pontos 3D em algumas técnicas. É necessário evitar essas interferências.

•Integração de hardware e software com outros sistemas. A câmara será controlada automaticamente pela unidade de controle central do próprio robô ou por uma fonte externa. Desenvolvimentos para esta finalidade são orientados para a integração e, atualmente, a

maioria dos sistemas de visão comerciais também está preparada para ser conectada a um robô e controlada por software externo usando bibliotecas.

•Orçamento. Fora das questões técnicas, para uma implementação real, o orçamento também deve ser considerado. Um equilíbrio entre custo e desempenho é necessário, pois a maioria das características anteriores pode ser alcançada ou aprimorada aumentando o valor investido [20].

Tabela 3- Comparação das vantagens e desvantagens das técnicas de visão artificial

	Vantagens	Desvantagens
Visão estéreo e fotogrametria	Comumente utilizado. Exatidão	Influenciado pelo ambiente. São necessárias marcas físicas A densidade da nuvem de pontos pode ser baixa O objeto e a câmara devem estar estáticos para a captura
Time of Flight (ToF)	Não são necessários marcos físicos	Influenciado pelo ambiente O objeto e a câmara devem estar estáticos para a captura
Sensor de Luz Estruturada	Exatidão	Por vezes influenciado pela luz ambiente Problemas para criar o modelo 3D para superfícies de determinadas cores Caro Os sensores podem ser bastante grandes. O objeto e a câmara devem estar estáticos para a captura
Codificação Luminosa	Económica Não é necessário que o objeto e a câmara permaneçam estáticos durante a captação	Baixa precisão Não certificada a nível industrial
Triangulação a Laser		Perigoso para as pessoas (dependendo da potência do laser)

	Comumente utilizado. Barato (dependendo da precisão do laser)	Normalmente, a distância de trabalho é curta.

2.3.7. *Deep Learning* em Visão Artificial

O *deep learning*, uma subárea do *machine learning*, utiliza *convolutional neural network* (CNN) para a análise e interpretação de dados visuais, destacando-se pela sua eficácia em tarefas como deteção, classificação e segmentação de objetos [28].

Uma CNN é uma rede neural projetada para processar dados visuais. A CNN é um tipo de modelo de *deep learning* amplamente utilizado numa grande variedade de aplicações de visão artificial onde é necessário reconhecer e classificar objetos de forma precisa e eficiente. Isto porque consegue aprender e identificar características importantes das imagens automaticamente [29].

O YOLO (You Only Look Once) é uma abordagem popular para deteção de objetos, processando toda a imagem em uma única passagem, dividindo-a numa grade e identificando objetos e as localizações em tempo real. A versão mais recente, YOLOv8, oferece melhorias significativas em termos de precisão e velocidade, tornando-se uma ferramenta crucial para a automação logística. Estas técnicas de *deep learning* melhoram significativamente a precisão e a eficiência de deteção e classificação de objetos, reduzindo erros e aumentando a eficiência operacional. Modelos como o YOLO permitem a automação de processos em tempo real. Além disso, os sistemas de *deep learning* podem ser treinados para se adaptarem a novas condições e tipos de produtos, oferecendo flexibilidade às operações logísticas. No entanto, o processo de treino de redes neurais profundas requer grandes quantidades de dados rotulados, o que pode ser um desafio em termos de coleta e anotação de dados. Além disso, essas redes requerem grande capacidade computacional tanto para o treino quanto para a inferência, implicando altos custos de implementação [28].

2.4. Estado da Arte

Referência Bibliográfica	Descrição do trabalho
[30]	O documento "Analysis and Observations From the First Amazon Picking Challenge" analisa as técnicas e estratégias utilizadas pelas equipes participantes do primeiro *Amazon Picking Challenge*. O evento foi criado para incentivar o desenvolvimento de soluções robóticas para tarefas de *pick and place* em armazéns. As equipas enfrentaram desafios como a manipulação de objetos de diferentes formas, tamanhos e materiais, e a integração de sistemas de visão artificial precisos e eficientes. Entre as técnicas utilizadas, destacam-se o uso de algoritmos de visão artificial para deteção e reconhecimento de objetos, como o YOLO (You Only Look Once), e a combinação de *machine learning* para melhorar a precisão e eficiência das operações. Embora tenham sido feitos progressos significativos, ainda há obstáculos consideráveis para alcançar uma automação robusta e eficaz em ambientes logísticos complexos.
[31]	O artigo "Pick and Place Robotic Arm: A Review Paper" oferece uma revisão abrangente sobre os braços robóticos de *pick and place*, abordando tecnologias de controle, sensores de feedback e integração com sistemas de visão artificial. As técnicas discutidas incluem o uso de algoritmos de planeamento de movimento e estratégias de controle baseadas em *machine learning*. O artigo destaca como a combinação dessas tecnologias pode melhorar a precisão, velocidade e adaptabilidade deste tipo de sistemas. Além disso, o uso de modelos de deteção de objetos como o YOLO é discutido como uma ferramenta eficaz para melhorar a precisão das operações.
[32]	Este documento retrata a implementação de um braço robótico para *pick and place* para a gestão de produtos em armazéns, utilizando técnicas de visão artificial baseadas em *deep learning*, incluindo o uso de modelos como o YOLO para deteção de objetos. Foram utilizados sensores de visão para detetar e classificar produtos, e algoritmos de controle para manipulação precisa. A integração desses componentes resultou em um sistema automatizado capaz de aumentar a produtividade e reduzir erros nas operações. O estudo também discutiu os desafios enfrentados durante a implementação, como a necessidade de grandes conjuntos de dados para treino dos modelos e a complexidade computacional envolvida. Os resultados mostraram uma melhoria significativa na eficiência operacional e na precisão das tarefas de *picking*, evidenciando o potencial das tecnologias de visão artificial na automação de armazéns.

[33]	O artigo "Vision Assisted Pick and Place Robotic Machine" discute o design e a implementação de uma máquina robótica de *pick and place* assistida por visão para atividades de laboratório. Utilizaram-se câmaras de alta resolução e algoritmos de visão artificial, incluindo o YOLO, para guiar o robô na execução de tarefas de *pick and place*. O sistema foi projetado para melhorar a precisão e a adaptabilidade das operações de *picking* em ambientes variados, utilizando técnicas de deteção de objetos para identificar e manipular itens com precisão. A pesquisa também abordou a importância da calibração das câmaras e do ajuste dos algoritmos de visão para diferentes condições de iluminação e tipos de objetos. Os ensaios realizados demonstraram que a visão artificial pode aumentar significativamente a eficácia das operações de *pick and place*, reduzindo o tempo de ciclo e aumentando a taxa de sucesso na manipulação de itens diversos.
[34]	Este documento aborda a visão robótica para reconhecimento de objetos e a sua posição para operações de *pick and place*. Foram utilizados algoritmos de visão artificial, como o YOLO, para detetar a posição e a rotação dos objetos, permitindo ao robô ajustar o seu posicionamento conforme a orientação detetada. Isso melhorou a precisão e a eficácia das operações robóticas, especialmente em cenários onde a manipulação precisa é crucial. A pesquisa também aborda a dificuldade de calibrar os algoritmos de visão para diferentes tipos de objetos e condições de iluminação. Os resultados ensaios mostraram que a capacidade de reconhecer a rotação dos objetos aumentou significativamente a eficiência das operações de *pick and place*, reduzindo o tempo de ajuste e aumentando a precisão na manipulação.

3. Métodos e Aplicação

O desenvolvimento focou-se na criação de modelos de visão artificial utilizando a plataforma Roboflow e na implementação de códigos em *Python* para deteção de objetos e *keypoints*. Nos próximos pontos vão ser explicadas, em detalhe, cada uma das etapas do processo.

3.1. Simulação de um Processo Logístico

Para simular um processo logístico, foi utilizado um conjunto de cinco paralelepípedos. Esses paralelepípedos foram selecionados para representar cargas de diferentes cores e foram utilizados para criar uma simulação de paletização, um processo comum em ambientes logísticos. A paletização envolve a organização e empilhamento de itens em uma palete de forma eficiente, maximizando o uso do espaço e garantindo a estabilidade da carga durante o transporte.

Para simular um processo logístico, foram utilizados cinco paralelepípedos, todos com dimensões iguais de 75mm x 25mm x 15mm. Esses paralelepípedos são feitos de madeira e possuem uma distribuição de peso homogênea, o que significa que o centro geométrico coincide com o centro de massa. Cada paralelepípedo tem uma cor distinta: amarelo, azul, laranja, verde e roxo com a cor sendo uniforme e homogênea em cada um deles. É possível verificar os paralelepípedos na Figura *16*.

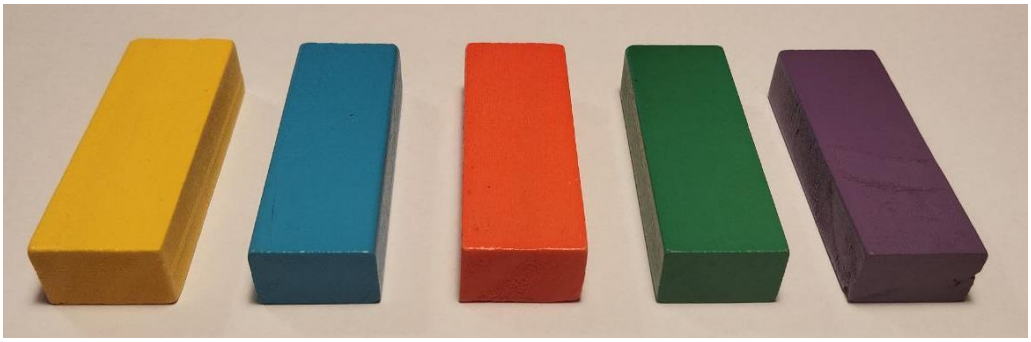


Figura 16- Paralelepípedos usados para simular o processo logístico

3.2. Ferramentas e Tecnologias Utilizadas

Para o desenvolvimento do sistema de visão, foram utilizadas diversas ferramentas, cada uma com um papel específico no processo. Nos próximos pontos são descritas as principais ferramentas e tecnologias utilizadas.

3.2.1. Python

O *Python* é uma linguagem de programação de alto nível amplamente utilizada em *data science*, *web development* e, especialmente, em *machine learning* e inteligência artificial. O *Python* é conhecido por usar uma sintaxe simples e legível, o que torna a programação mais acessível e menos propensa a erros. Esta linguagem suporta várias bibliotecas poderosas, como *Numpy*, *TensorFlow* e *PyTorch*, que são essenciais para a construção e implementação de modelos de *machine learning*. Além disso, *Python* possui uma grande comunidade ativa, que oferece uma ampla gama de recursos e suporte para os seus utilizadores [35][36].

O *Python* permite a rápida prototipagem e desenvolvimento de algoritmos complexos. Por exemplo, o uso de bibliotecas como *Numpy* facilita a manipulação de grandes conjuntos de dados e a realização de operações matemáticas necessárias para calcular o centro de massa dos paralelepípedos. Além disso, a vasta documentação e os recursos disponíveis online ajudam a resolver problemas e implementar novas funcionalidades de forma eficiente [35][36].

3.2.2. Roboflow

O *Roboflow* é uma plataforma que trabalha maioritariamente em *cloud* para fazer a gestão de fluxo de dados, especialmente desenvolvida para robôs que utilizam inteligência artificial. Esta plataforma facilita o desenvolvimento de sistemas robóticos, dividindo o processo em quatro módulos principais: processamento de dados, desenvolvimento de algoritmos, testes e adaptação de aplicações. Cada módulo interage com um motor de dados centralizado, permitindo uma alta reusabilidade e manutenção dos componentes [34][35].

Ao usar *Roboflow*, é possível anotar rapidamente as imagens de paralelepípedos e treinar modelos de deteção de objetos de forma eficiente. A plataforma automatiza grande parte do processo de pré-processamento de dados, como a normalização de imagens e a criação de conjuntos de treinamento e validação, economizando tempo e recursos [37].

3.2.3. YOLOv8

O *YOLO* (*You Only Look Once*) é uma família de modelos de deteção de objetos em tempo real que têm sido continuamente melhorados ao longo dos anos. A versão *YOLOv8* é notória pela sua precisão e velocidade, permitindo a deteção de vários objetos numa única passagem. A arquitetura do *YOLOv8* permite-lhe realizar previsões rápidas, mantendo um elevado nível de precisão, o que o torna ideal para aplicações que requerem processamento em tempo real [34][36].

O *YOLOv8* consegue identificar múltiplos paralelepípedos numa passagem pela imagem, fornecendo coordenadas precisas em milissegundos. Isso permite que o sistema reaja rapidamente a mudanças na configuração dos paralelepípedos, garantindo uma paletização eficiente e estável [39].

3.2.4. Visual Studio Code

Visual Studio Code (*VS Code*) é uma ferramenta robusta, facilmente configurável e altamente funcional, projetada tanto para iniciantes quanto para programadores experientes. Este Ambiente de Desenvolvimento Integrado (*IDE*) suporta a criação e edição de projetos em várias linguagens de programação, como *C#*, *C++*, *Clojure*, *F#*, *HTML*, *Java*, *Lua*, *PHP*, *Perl*, *Python*, *SQL*, *Visual Basic* e *XML*. O *VS Code* oferece recursos de conclusão de código, *snippets* de código, suporte para fazer e corrigir código, além de integração com o controle de versão *Git*.

3.2.5. Câmaras Utilizadas

As fotos usadas para anotações e treino dos modelos foram tiradas com a câmara do telefone *Xiaomi Note 12*, que possui uma câmara principal de 50MP com abertura de f/1.8. Essa câmara foi escolhida pela sua alta resolução e qualidade de imagem, garantindo detalhes nítidos necessários para a anotação precisa dos dados.

A câmara que será usada no projeto final é a Intel RealSense D435i (*Figura 17*), cujo *datasheet* é fornecido no Anexo A, é uma câmara de profundidade estéreo que combina sensores de imagem com um projetor infravermelho para capturar dados de profundidade de alta precisão. As especificações incluem resolução de até 1280 x 720 para profundidade, até 1920 x 1080 para RGB, campo de visão diagonal de mais de 90 graus e suporte para *streaming* de profundidade a até 90 FPS. A D435i também inclui uma Unidade de Medição Inercial (IMU) para dados de 6 graus de liberdade (6DoF). A câmara é ideal para aplicações que requerem dados de profundidade precisos e integridade estrutural.

Figura 17- Câmara Intel RealSense Depth D435i

3.3. Recolha e Anotação de Dados

3.3.1. Processo de Recolha de Dados

Para treinar os modelos de identificação de objetos e de *keypoints*, foi necessário recolher um conjunto abrangente de imagens dos paralelepípedos em várias posições e condições de iluminação. As etapas específicas do processo de coleta de dados foram as seguintes:

Configuração da Câmara: As fotos foram tiradas com a câmara do Xiaomi Note 12, que possui uma resolução de 50MP e abertura de f/1.8, garantindo imagens de alta qualidade e detalhadas.

Ambiente Controlado: As imagens foram capturadas em um ambiente controlado para assegurar a consistência dos dados. A iluminação foi ajustada para minimizar sombras e reflexos, como demonstrado na *Figura 18*.

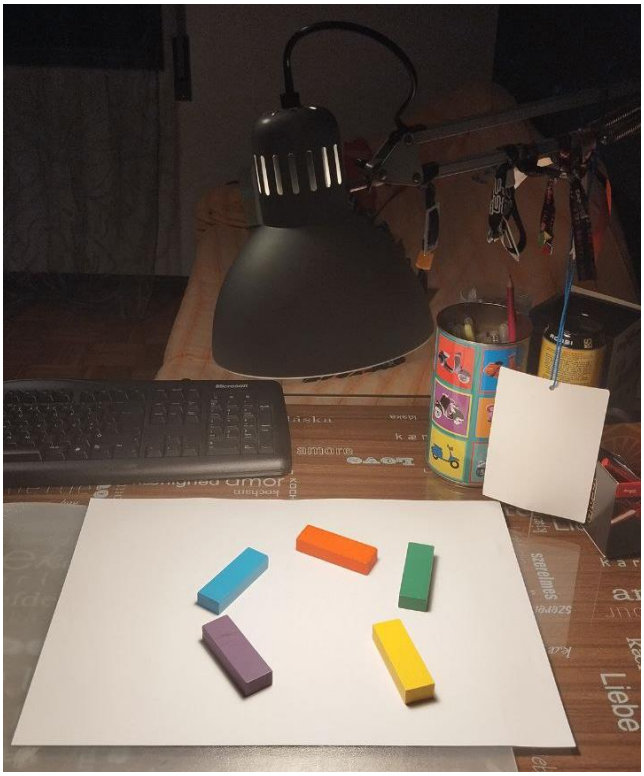


Figura 18- Ambiente controlado para recolha de dados

Diversidade de Posições: Os paralelepípedos foram fotografados em diferentes posições e ângulos para criar um conjunto de dados diversificado, essencial para treinar um modelo robusto.

Número de Imagens: Foram recolhidas um total de duzentas imagens. A escolha deste número baseou-se na simplicidade e simetria dos objetos, o que reduz a necessidade de um grande volume de dados para um treinamento eficaz.

Das 200 imagens recolhidas, cem imagens são com apenas os cinco blocos. Estas imagens fornecem dados limpos e claros, permitindo que o modelo aprenda as características fundamentais dos paralelepípedos sem interferência de outros objetos, *Figura 19*.

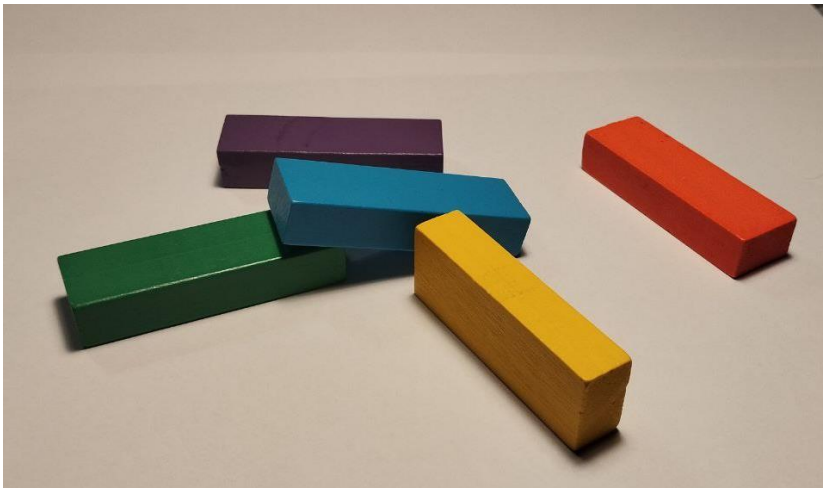


Figura 19- Exemplo de uma imagem recolhida apenas com os blocos

As outras 100 imagem contém mais objetos, a inclusão de outros objetos nas imagens visa aumentar a robustez do modelo. Essas imagens simulam um ambiente logístico mais realista, onde os blocos podem estar misturados com outros itens. Esta diversidade ajuda o modelo a generalizar melhor em situações do mundo real*, Figura 20*.

Figura 20- Exemplo de uma imagem recolhida com os blocos e outros itens

3.3.2. Processo de Anotação de Dados

Após a recolha das imagens, o próximo passo foi a anotação dos dados, onde cada imagem foi rotulada para incluir as informações necessárias para treinar os modelos. As imagens recolhidas foram carregadas na plataforma *Roboflow*, que oferece ferramentas intuitivas para anotação de dados, como se verifica na *Figura 21*.

Cada paralelepípedo nas imagens foi anotado manualmente, e as classes criadas para cada modelo foram as seguintes: Modelo de Identificação de Blocos: Bloco Amarelo, Bloco Azul,

Bloco Laranja, Bloco Roxo, Bloco Verde. Modelo de Identificação dos *Keypoints*: Bloco Amarelo KP, Bloco-Azul-KP, Bloco-Laranja-KP, Bloco-Roxo-KP, Bloco-Verde-KP.

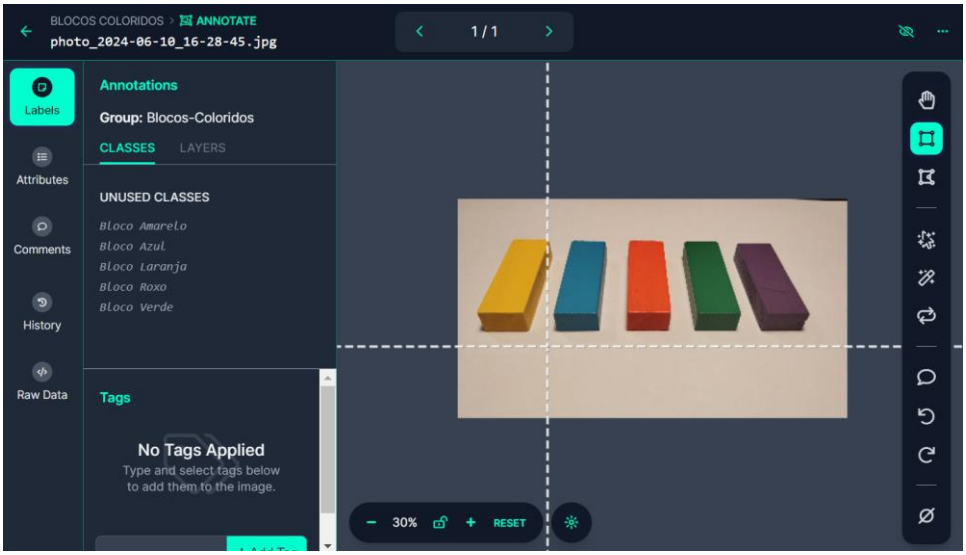


Figura 21- *Display* de anotação da *Roboflow*

No caso do modelo de identificação dos *keypoints*, cada classe possuía os *keypoints* anotados numa estrutura conforme ilustrada na Figura 22, com os seguintes vértices: Vertice-BA, Vertice-BAZ, Vertice-BL, Vertice-BR, Vertice-BV.

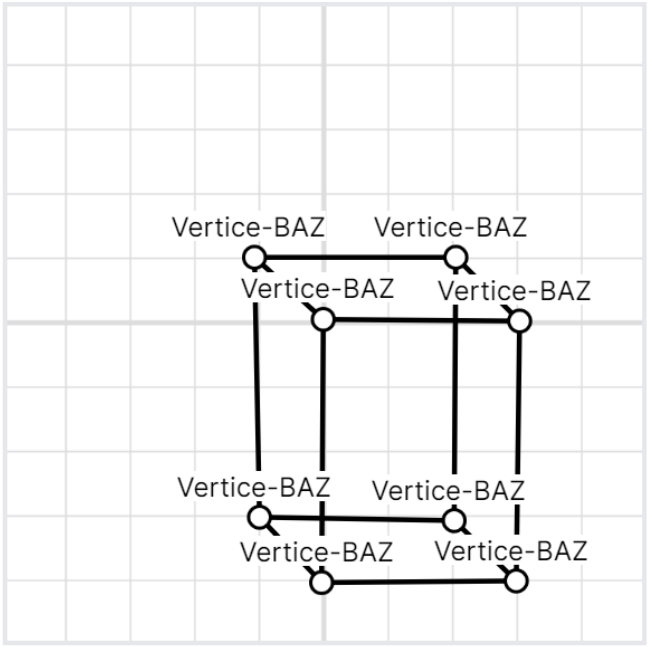


Figura 22- Estrutura de Anotações dos Keypoints do Bloco-Azul-KP

Para o modelo de identificação de objetos, as caixas delimitadoras foram desenhadas de forma justa ao redor de cada paralelepípedo (*Figura* 23). Essa precisão é esperada para aumentar a precisão do modelo, pois reduz o ruído e foca exatamente nos contornos dos objetos.

Figura 23- Imagem anotada do *dataset* do modelo - Identificação de blocos

No caso do modelo Identificação dos *keypoints*, as caixas de anotação não permitiram o contorno preciso da peça, sendo apenas possível fazer retângulos. Isso simplifica o processo de anotação e foca nos pontos principais (vértices), como é visível na Figura *24*, embora seja previsível que possa introduzir algum grau de imprecisão.

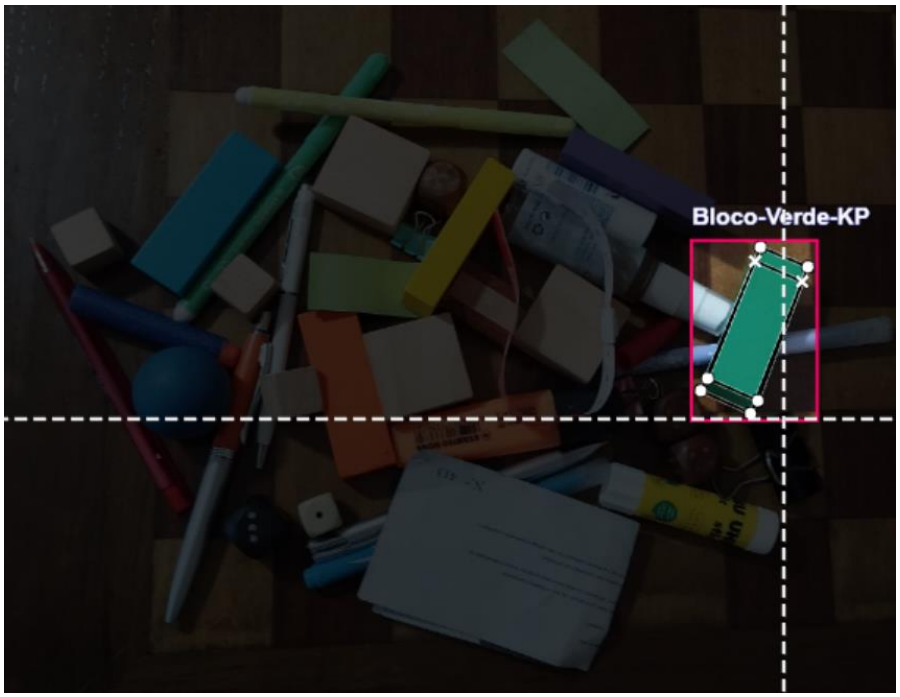


Figura 24- Imagem anotada do *dataset* do modelo - Identificação dos *keypoints*

Para ilustrar o processo de anotação, consideramos uma imagem de exemplo onde os paralelepípedos são identificados e rotulados. Na identificação de objetos, as caixas

delimitadoras são desenhadas ao redor de cada paralelepípedo, ajustadas precisamente aos contornos dos blocos. Na identificação de *keypoints*, os vértices de cada paralelepípedo são marcados com pontos específicos dentro de retângulos, que posteriormente serão usados para calcular o centro de massa.

Escolher os vértices como *keypoints* para os paralelepípedos foi uma decisão estratégica tendo várias considerações. Em primeiro lugar, os vértices de um paralelepípedo são pontos claramente definidos e fáceis de identificar, independentemente da orientação do objeto, o que garante simplicidade geométrica no processo de anotação e reduz a complexidade do modelo [40].

Comparando a outros pontos possíveis, como bordas ou superfícies, os vértices são uniformemente distribuídos e apresentam características únicas que garantem uniformidade e os tornam facilmente distinguíveis pelo modelo. Os vértices também são invariantes a transformações comuns como rotação e escala, e são repetíveis em diferentes instâncias do mesmo objeto, garantindo invariância e repetibilidade, independentemente das variações na orientação ou tamanho do objeto, como se verifica na Figura *25*, assegurando a completude da anotação [40].

Outra vantagem é que, por haver mais de um vértice por bloco e como todos os vértices do mesmo bloco foram anotados com o mesmo nome, garantindo maior eficiência durante o treino do modelo devido à repetição consistente desses pontos.

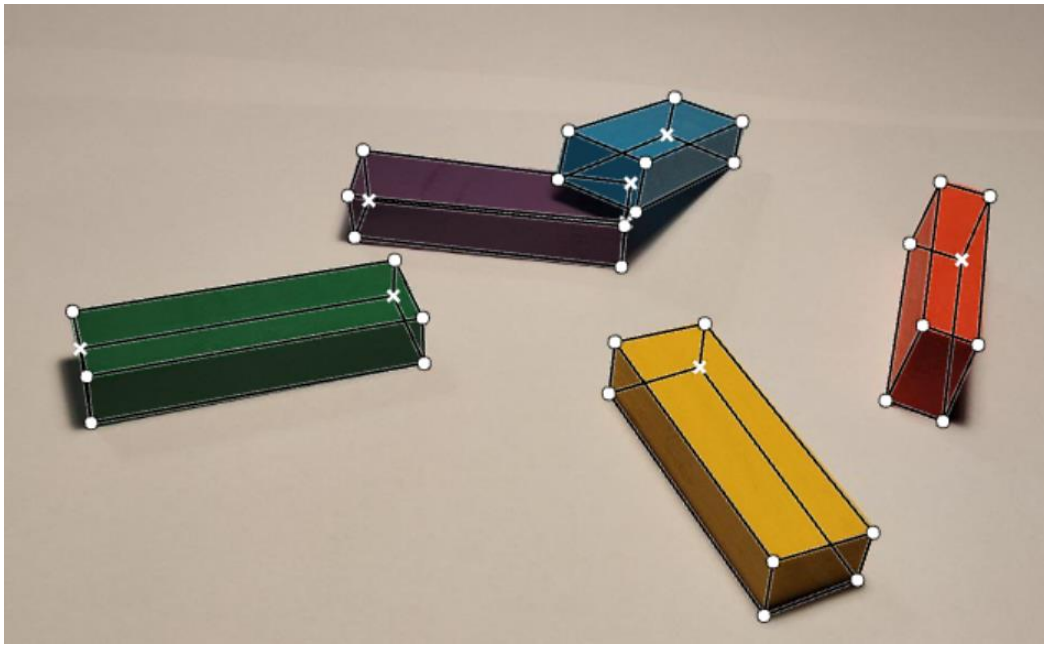


Figura 25- Anotação dos *keypoints* numa imagem do *dataset*

A anotação manual, apesar de ser um processo demorado, oferece várias vantagens. Garante que os dados estejam anotados com alta precisão, o que é fundamental para o treino eficaz dos modelos. Além disso, a anotação cuidadosa assegura que o conjunto de dados tenha alta qualidade, resultando em um modelo mais robusto e preciso. Finalmente, permite ajustes e melhorias contínuas, garantindo que o modelo possa ser refinado com base nos resultados.

Ao seguir esse processo detalhado de recolha e anotação de dados, garantiu-se que os modelos de visão artificial foram treinados com dados de alta qualidade, aumentando a precisão e a robustez das previsões realizadas durante a simulação do processo logístico.

3.4. Treino dos Modelos

3.4.1. Preparação dos Dados para o Treino

Os dados anotados foram divididos em três subconjuntos: treino, validação e teste. A divisão foi feita da seguinte forma: setenta por cento (70%) dos dados foram destinados ao treinamento, vinte por cento (20%) para validação e dez por cento (10%) para teste, *Figura 26*. Essa divisão foi escolhida com base nas recomendações da plataforma *Roboflow*, que sugere esses valores como ideais para um equilíbrio eficiente entre treino, validação e teste de modelos de visão artificial [41].

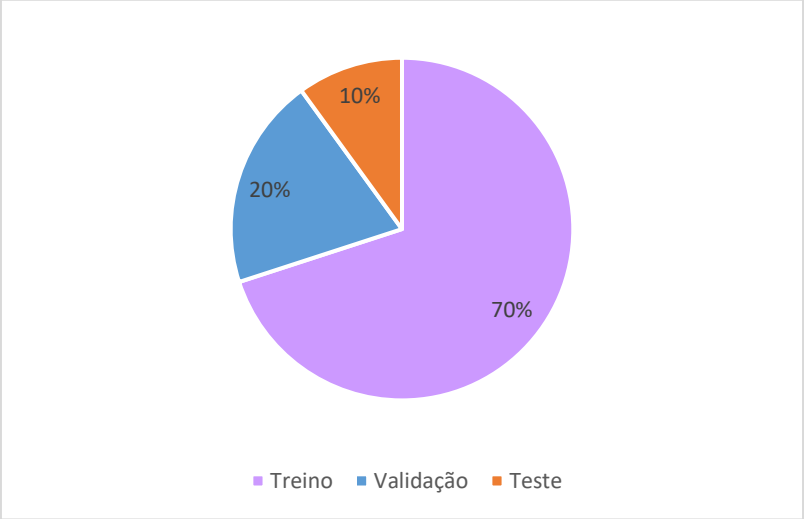


Figura 26- Divisão dos dados em três subconjuntos

A divisão de 70/20/10 foi escolhida para garantir um treino adequado, validação confiável e teste abrangente do modelo. O grande subconjunto de dados de treino (70%) permite que o modelo aprenda características variadas e complexas dos dados, resultando em um modelo mais robusto. A validação (20%) é usada para monitorizar o desempenho do modelo durante o treino e ajustar hiperparâmetros, prevenindo o *overfitting*, que ocorre quando o modelo se ajusta precisamente aos dados do treino, provocando imprecisões ao analisar dados novos. O teste (10%) é utilizado para avaliar o desempenho final do modelo em dados completamente novos, fornecendo uma estimativa precisa de seu desempenho em cenários reais[42][43].

A escolha correta dos hiperparâmetros, como a taxa de aprendizagem e a dimensão do *batch*, é crucial para o desempenho do modelo, porque influencia diretamente a capacidade do modelo de se adaptar para novos dados [43].

Essa abordagem de divisão dos dados foi aplicada para ambos os modelos, tanto o de identificação de blocos quanto o de identificação dos *keypoints*, garantindo consistência no processo de treino, avaliação e teste.

3.4.2. Pré-processamento e Reforço dos Dados

Após a divisão do conjunto de dados, o próximo passo é o pré-processamento e o reforço dos dados. O objetivo deste passo é preparar e melhorar as imagens para o treino dos modelos, garantindo que os dados sejam consistentes e diversificados. Isso ajuda a prevenir o *overfitting* e melhora a capacidade do modelo de generalizar para novas imagens.

O *Roboflow* adiciona automaticamente dois passos ao pré-processamento: auto-orientação e redimensionamento 640x640. A auto-orientação assegura que as imagens são armazenadas no disco da mesma forma que os aplicativos as abrem. O redimensionamento cria um tamanho consistente para as imagens, facilitando e agilizando o processo de treino. Para ambos os modelos, foi escolhido utilizar auto-orientação e redimensionamento como etapas de pré-processamento (Figura *27*).

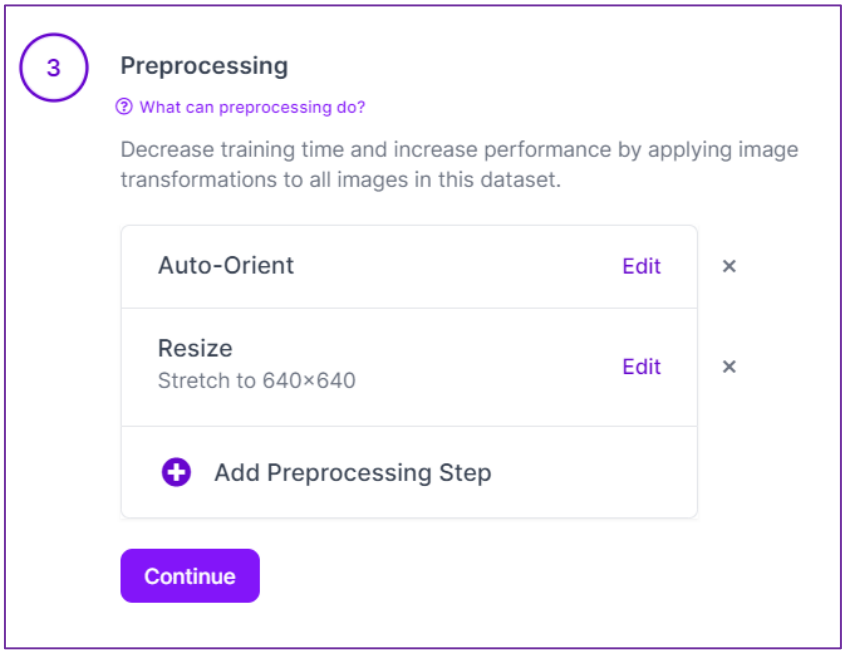


Figura 27- Passos de pré-processamento selecionados no Roboflow

O reforço de dados gera várias variações de cada imagem original para ajudar o modelo a generalizar melhor.

Identificação de Blocos: Foi utilizada a técnica de *flip* aleatório. As imagens são espelhadas horizontalmente e verticalmente para aumentar a variabilidade dos dados e ajudar o modelo a identificar os paralelepípedos independentemente da sua orientação. O *flip* é particularmente útil porque os blocos podem aparecer em qualquer orientação, e ajuda o modelo a aprender a identificar os blocos de diferentes ângulos (Figura *28*).

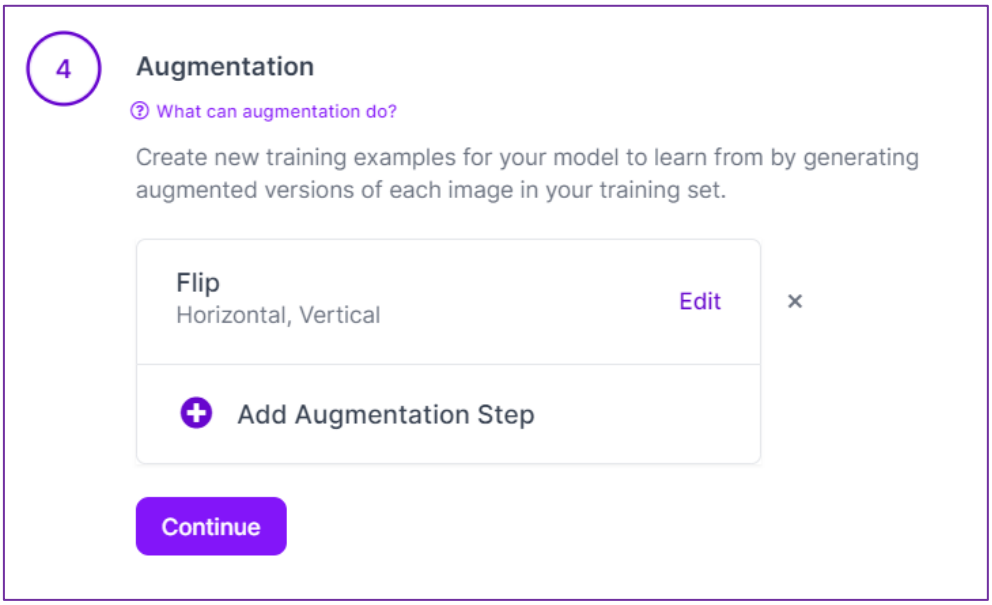


Figura 28- Passo para reforço do modelo – Identificação de Blocos

Deteção de *Keypoints*: Foi utilizada a modificação de brilho (*brightness* +/- 15%). As imagens têm o seu brilho ajustado aleatoriamente para melhorar a robustez do modelo contra diferentes condições de iluminação. A modificação de brilho é importante porque os *keypoints* precisam de ser detetados com precisão mesmo que as condições de iluminação não sejam exatamente iguais ao *dataset,* como se verifica na Figura *29*.

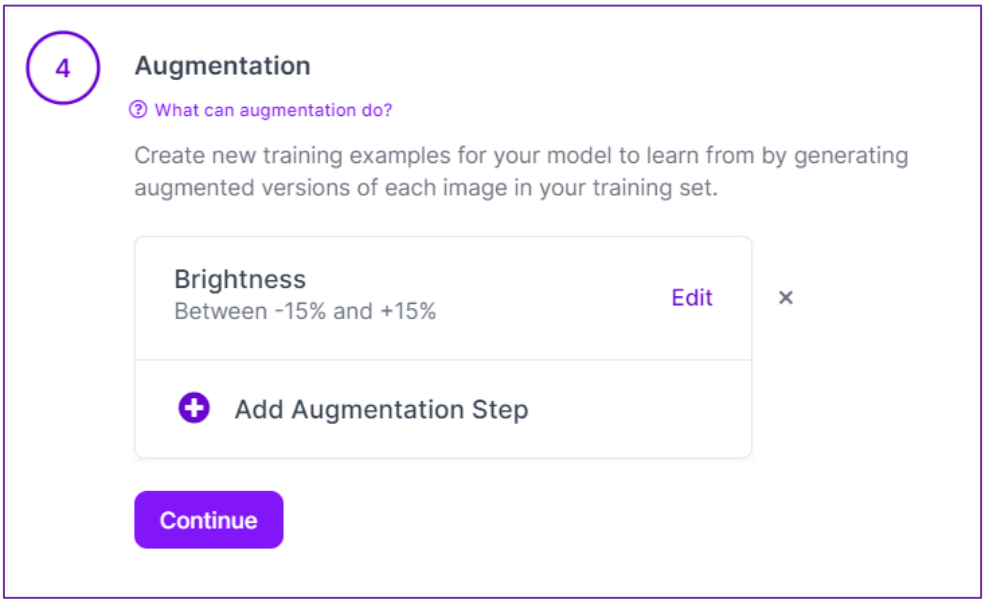


Figura 29- Passo para reforço do modelo – Identificação dos Keypoints

Estas técnicas de reforço foram escolhidas porque ajudam os modelos a aprender características robustas dos objetos, tornando-os menos sensíveis a variações na orientação e nas condições de iluminação. A robustez nas características permite que o modelo mantenha um desempenho elevado e confiável em diferentes cenários do mundo real.

3.4.3. Gestão do Ambiente de Treino pela Roboflow

A *Roboflow* é uma plataforma que facilita o treino de modelos de visão artificial, gerindo automaticamente o ambiente de treinamento. Isso inclui a configuração de hardware e software, bem como a gestão de custos associados ao uso de GPUs (*Graphics Processing Unit*).

Ao utilizar a *Roboflow*, todo o processo de configuração de hardware e software é automatizado. Isso elimina a necessidade de configurar manualmente máquinas virtuais, instalar drivers de GPU ou ajustar configurações específicas do sistema operacional. Com isso, o tempo necessário para iniciar o treino do modelo é significativamente reduzido.

Com a gestão automatizada do ambiente de treino, possibilitou-se o foco na otimização dos modelos para alcançar os melhores resultados possíveis. Os dados de desempenho e a análise detalhada dos resultados serão discutidos no ponto Resultados e Discussão.

3.5. Código e Funcionamento

3.5.1. Código de Identificação de Blocos

O código para identificação de blocos foi desenvolvido para identificar e localizar os paralelepípedos coloridos nas imagens capturadas em tempo real. Utilizando o modelo YOLOv8, o código processa as imagens de entrada, realiza a inferência para detetar os objetos e exibe as imagens anotadas com caixas delimitadoras ao redor dos blocos detetados. Além disso, o código indica o nível de confiança de cada deteção e imprime no terminal as etiquetas dos objetos reconhecidos. O objetivo é detetar a presença e a localização dos blocos nas imagens com alta precisão.

Resumindo, o código no Apêndice C, configura um pipeline de inferência para a deteção dos blocos, processa as previsões extraindo e exibindo informações relevantes, anota os quadros de vídeo com os blocos detetados e os níveis de confiança, e exibe o vídeo anotado em tempo real.

Começa por importar os módulos e bibliotecas necessários para configurar o pipeline de deteção de objetos. Importa a classe InferencePipeline e a classe VideoFrame de um módulo personalizado inference, que são essenciais para gerir o processo de inferência e os frames de vídeo. Também importa a biblioteca OpenCV (cv2) para exibir imagens anotadas e a biblioteca supervision (sv) para visualizar e anotar as previsões. Estas importações são fundamentais para permitir que o pipeline processe os frames do vídeo, execute a deteção de objetos e visualize os resultados.

De seguida define e implementa uma função personalizada (my_custom_sink) que processa as previsões feitas pelo modelo de inferência. Primeiro, cria um anotador de caixas

(BoxAnnotator) da biblioteca supervision para ser usado na anotação dos objetos detetados. A função my_custom_sink recebe as previsões (predictions) e os frames do vídeo (video_frame). Extrai os rótulos de texto para cada previsão, incluindo a classe e o nível de confiança, e imprime-os na consola. Em seguida, carrega as previsões no objeto *Detections* usando *sv.Detections.from_inference*. O *frame* de vídeo é anotado com as deteções e rótulos utilizando o anotador do *supervision*, e a imagem anotada é exibida numa janela *OpenCV* denominada "*Predictions*". Finalmente, a função *cv2.waitKey(1)* espera 1 milissegundo para permitir a atualização da janela de visualização.

De seguida inicializa e inicia o *InferencePipeline* para deteção de objetos. O método *InferencePipeline.init* é chamado com vários parâmetros: *model_id* especifica o identificador do modelo a ser usado, *video_reference* define a fonte de vídeo (0 refere-se à webcam, que serviu para testar o código), *on_prediction* atribui a função personalizada de *sink* (my_custom_sink) para processar as previsões, e *confidence* define o limiar de confiança para as previsões em 90%. Após a inicialização do *pipeline*, *pipeline.start()* começa o processo de inferência, e *pipeline.join()* aguarda a conclusão do processo, garantindo que a *thread* principal espere pela finalização da *thread* de inferência.

Para executar o código é necessário definir a API Key do modelo do Roboflow na linha de comandos utilizando o comando "set ROBOFLOW_API_KEY=**********". Esta API Key não será revelada, pois ela garante acesso ao dataset, permitindo alterá-lo.

Para executar o código deve ser usado o comando "python nome_do_programa.py". O código capturará frames de vídeo da câmera (referida por video_reference), realizará a inferência para detetar os paralelepípedos e exibirá as imagens anotadas com as caixas delimitadoras ao redor dos objetos detetados. Além disso, as etiquetas dos objetos reconhecidos, juntamente com seus níveis de confiança, serão apresentadas no terminal. Ao rodar o código, uma janela será aberta mostrando o vídeo em tempo real com as deteções.

3.5.2. Código de Identificação dos Keypoints

O código de deteção de *keypoints* foi desenvolvido para identificar os vértices dos paralelepípedos nas imagens. Utilizando um modelo de deteção de *keypoints*, o código processa as imagens capturadas, realiza a inferência para detetar os vértices dos objetos e exibe as imagens anotadas com os *keypoints* e o cálculo do centro de massa. Além disso, o código imprime no terminal as coordenadas dos vértices e os centros de massa calculados. O objetivo é identificar com precisão os vértices dos paralelepípedos e calcular o centro de massa de cada bloco.

O código no Apêndice D está organizado em várias funções principais: o carregamento do modelo, que utiliza a classe *InferencePipeline* para carregar o modelo de deteção de *keypoints*; o pré-processamento da imagem, que prepara a imagem capturada pela câmara antes da inferência; a inferência, que executa a deteção de *keypoints* na imagem; e o pós-processamento, que processa os resultados da deteção, desenha os *keypoints*, calcula o centro de massa, exibe as imagens anotadas e apresenta as coordenadas dos *keypoints* e centros de massa no terminal.

Inicia por importar as bibliotecas e módulos necessários para configurar o *pipeline* de inferência para deteção de objetos e análise de pontos-chave. Especificamente, importa *InferencePipeline* e *VideoFrame* de um módulo personalizado *inference*, que são utilizados para gerir o processo de inferência e os *frames* de vídeo, respetivamente. Também importa a biblioteca *OpenCV* (cv2) para processamento de imagem e vídeo, a biblioteca *NumPy* (np) para operações numéricas, e a biblioteca *supervision* (sv) para visualizar e anotar as previsões. Estas importações são fundamentais para que as partes subsequentes do código funcionem corretamente, permitindo o processamento, análise e visualização dos *frames* de vídeo em tempo real.

De seguida, define uma função personalizada, *my_custom_sink*, que processa as previsões do modelo de inferência e anota os *frames* do vídeo. Começa por criar uma instância de *BoxAnnotator* da biblioteca *supervision*, que será usada para desenhar caixas delimitadoras em torno dos objetos detetados. A função *my_custom_sink* recebe dois argumentos: *predictions*, um dicionário que contém as previsões feitas pelo modelo, e *video_frame*, um objeto *VideoFrame* que representa o *frame* atual do vídeo. Extrai rótulos de texto para cada previsão, carrega as previsões na *API Detections* da biblioteca *supervision*, e usa o *box_annotator* para anotar o *frame* do vídeo com as previsões. O frame anotado é então armazenado na variável *image*.

De seguida, processa os *Keypoints* dos objetos detetados e calcula o seu centro de massa. Para cada previsão no dicionário *predictions*, verifica se existem pontos-chave. Se houver pontos-chave, extrai as suas coordenadas x e y e armazena-as numa matriz *NumPy*. Posteriormente, o centro de massa é calculado tirando a média das coordenadas dos *Keypoints* ao longo do eixo 0, o que corresponde a calcular a média das coordenadas x e y separadamente. Isto é feito usando a função *np.mean* do *NumPy*, onde *axis=0* especifica que a média deve ser calculada ao longo das linhas, efetivamente calculando a média das coordenadas *x* e *y* de todos os *keypoints*. O parâmetro *dtype=int* assegura que as coordenadas resultantes do centro de massa sejam inteiras, o que é adequado para coordenadas de pixels. A classe do objeto e as coordenadas do centro de massa são impressas na consola. O código desenha os *keypoints* e o centro de massa na imagem anotada usando a função *cv2.circle* do *OpenCV*. Finalmente, a imagem anotada é exibida numa janela chamada "*Predictions*", e o *display* é atualizado a cada 1 milissegundo com *cv2.waitKey(1)*.

Posteriormente inicializa e abre o *pipeline* de inferência para deteção de pontos-chave num fluxo de vídeo. O método *InferencePipeline.init* é chamado com vários parâmetros: *model_id* especifica o modelo a ser usado, video_reference define a fonte de vídeo (com 2 referindo-se à camara da Intel), *on_prediction* atribui a função personalizada *my_custom_sink* para processar as previsões, e *confidence* define o limiar de confiança para as previsões em 90%. Após a inicialização do *pipeline*, *pipeline.start()* começa o processo de inferência, e *pipeline.join()* assegura que a *thread* principal aguarde a conclusão do processo de inferência, executando o *pipeline* de inferência continuamente.

O código capturará *frames* do vídeo da câmara (referenciada por *video_reference*=2), realizará a inferência para detetar os *keypoints* e exibirá as imagens anotadas com os *keypoints*

desenhados e os centros de massa calculados. Além disso, os objetos anotados e as coordenadas dos centros de massa serão impressos no terminal. Ao rodar o script, uma janela será aberta mostrando os *frames* de vídeo em tempo real com as anotações.

3.5.3. Código de Integração dos Modelos

O código no Apêndice E foi desenvolvido para combinar as funcionalidades dos modelos de Identificação de Blocos e Identificação dos *Keypoints*, permitindo uma análise completa das imagens capturadas. Primeiramente, o modelo de deteção de objetos identifica e localiza os paralelepípedos nas imagens. Em seguida, o modelo de deteção de *keypoints* identifica os vértices desses objetos e calcula o centro de massa. O código exibe as imagens anotadas com as caixas delimitadoras dos objetos detetados e os centros de massa.

O código começa por importar as bibliotecas e módulos necessários para configurar o *pipeline* de inferência *multi-thread*. Importa o *OpenCV* (*cv2*) para processamento de imagem e vídeo, o *NumPy* (*np*) para operações numéricas, e classes de um módulo personalizado *inference* para gerir o processo de inferência (*InferencePipeline*) e *frames* de vídeo (*VideoFrame*). Também importa a biblioteca *supervision* (*sv*) para visualizar e anotar previsões, e o módulo *threading* para permitir *multi-threading*. Estas importações são essenciais para que o código subsequente funcione corretamente, permitindo o processamento, análise e visualização de *frames* de vídeo num ambiente *multi-thread*.

Seguidamente configura um anotador de caixas simples e define um estado compartilhado para armazenar previsões de pontos-chave e objetos usando um dicionário e um bloqueio de *threads*. A função *my_custom_sink_keypoints* processa previsões de *keypoints*. Começa por criar uma cópia do *frame* de vídeo para desenhar sobre ele. Para cada previsão que contém *keypoints*, extrai as coordenadas *x* e *y*, calcula o centro de massa ao fazer a média dessas coordenadas usando *NumPy*, e depois desenha um círculo no centro de massa na imagem copiada usando a função *cv2.circle* do *OpenCV*. Finalmente, a função retorna a imagem com o centro de massa desenhado.

De seguida a função *combined_sink* sincroniza e processa tanto as previsões de *Keypoints* quanto as previsões de objetos. Utiliza um bloqueio de *threads* para garantir acesso seguro ao estado compartilhado. Se as previsões de *Keypoints* estiverem disponíveis, elas são armazenadas no estado compartilhado. Da mesma forma, as previsões de objetos são armazenadas se disponíveis. Quando tanto as previsões de pontos-chave quanto as previsões de objetos estão presentes, chama *my_custom_sink_keypoints* e *my_custom_sink_object* para processar e anotar os *frames*. O estado compartilhado para previsões de *Keypoints* e de objetos é então redefinido para *None*.

De seguida inicializa e inicia dois *pipelines* de inferência separados, cada um executado na sua própria *thread*. O *pipeline_keypoints* é inicializado para lidar com a deteção dos vértices com o modelo e referência de vídeo especificados, utilizando a função *combined_sink* para processar as previsões. Da mesma forma, o *pipeline_object* é inicializado para lidar com a deteção de objetos. Ambos os pipelines definem um limiar de confiança de 90%. Para garantir que estes pipelines sejam executados simultaneamente*, keypoints_thread* e *object_thread* são criadas

usando a classe *threading.Thread*, cada uma direcionada para o método de início dos respetivos pipelines. Ambas as *threads* são então iniciadas com *start()* e unidas com *join()*, garantindo que a *thread* principal aguarde a conclusão de ambas as *threads*.

O código capturará *frames* de vídeo da câmara (referenciada por *video_reference*), realizará a inferência para detetar os blocos e os *keypoints*, exibindo as imagens anotadas com caixas delimitadoras e os centros de massa. As impressões dos resultados no terminal foram desativadas para otimizar o desempenho do código, uma vez que o programa estava apresentando lentidão significativa. Durante a execução do script, duas *threads* serão iniciadas para capturar e processar as imagens em tempo real, apresentando as deteções e centros de massa na janela de exibição, como exemplificado na Figura *30*.

Para facilitar a replicação dos resultados e a execução do programa desenvolvido, um guia passo a passo está disponível no Apêndice A, detalhando todas as etapas necessárias.

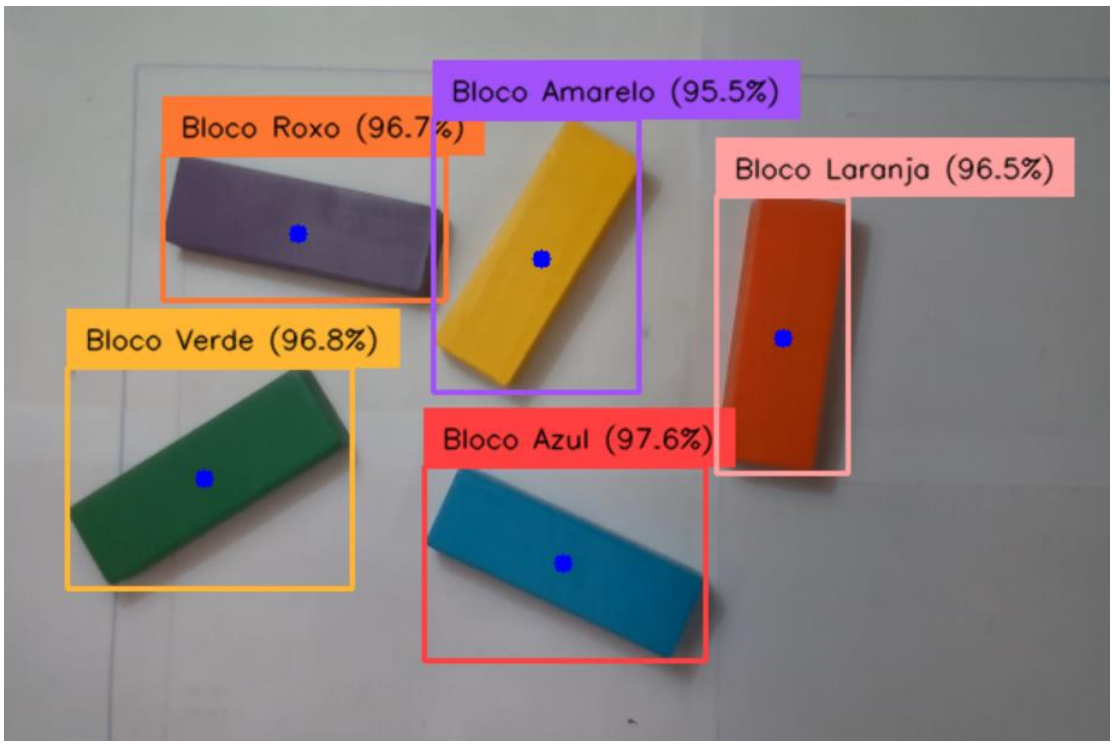


Figura 30- Display do Progrma de Integração dos Modelos

4. Resultados e Discussão

Neste capítulo, são apresentados e discutidos os resultados obtidos durante a execução do projeto. Primeiramente, detalhamos os resultados dos modelos de identificação de blocos e identificação de *keypoints* treinados com o *Roboflow*, incluindo métricas de precisão, *recall*, e análise de desempenho.

Comparamos os dois modelos, destacando que a deteção de objetos (blocos) apresentou uma precisão ligeiramente maior em relação à identificação de *keypoints*. Discutimos as possíveis razões para essa diferença, como a maior complexidade da tarefa de deteção de *keypoints* e o potencial aumento de ruído nas anotações de pontos específicos.

Além disso, abordamos a necessidade significativa de poder computacional para o treinamento dos modelos, que exigiu o uso intensivo de GPUs para processamento eficiente e a gestão de grandes volumes de dados.

Por fim, apresentamos a comparação entre os centros de massa calculados pelo programa e os valores reais, validando a precisão geométrica das deteções realizadas pelos modelos. A conclusão reforça a eficácia dos modelos e a robustez das metodologias utilizadas, confirmando a alta performance alcançada.

4.1. Apresentação de resultados

4.1.1. Resultados da Identificação de Blocos

Da Figura *31* pode-se obter os seguintes valores:
Número de imagens: 201
Número de anotações: 1016
Tamanho médio da imagem: 0.92 MP
Mediana da proporção da imagem: 1280x720

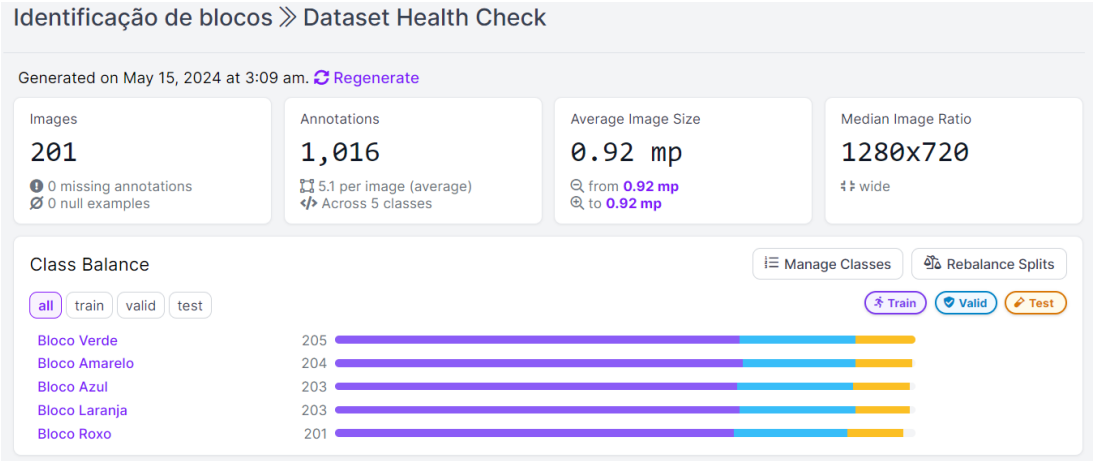


Figura 31- Dataset Health Check

A distribuição das dimensões das imagens, com a maioria das imagens tendo uma proporção de 1280x720 pixels, visível na Figura *32*.

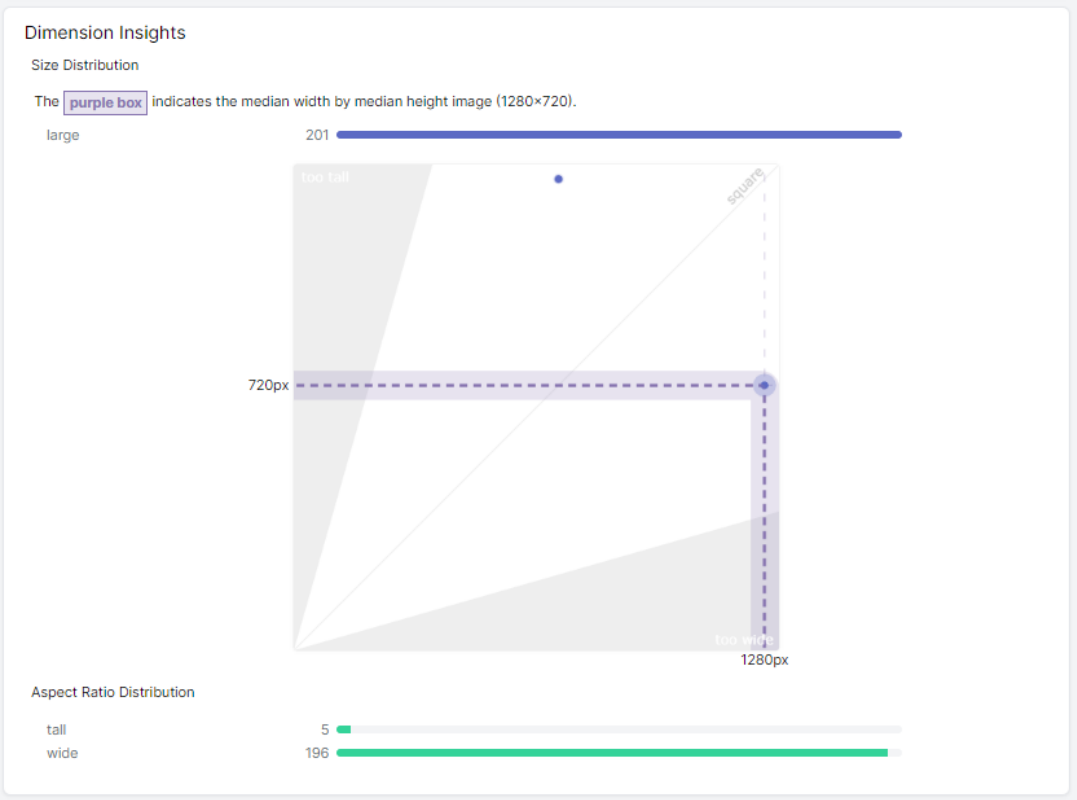


Figura 32- Distribuição do Tamanho

Na Figura *33* é possível verificar os mapas de calor das anotações feitas durante o desenvolvimento do *dataset* para o modelo de identificação de blocos.

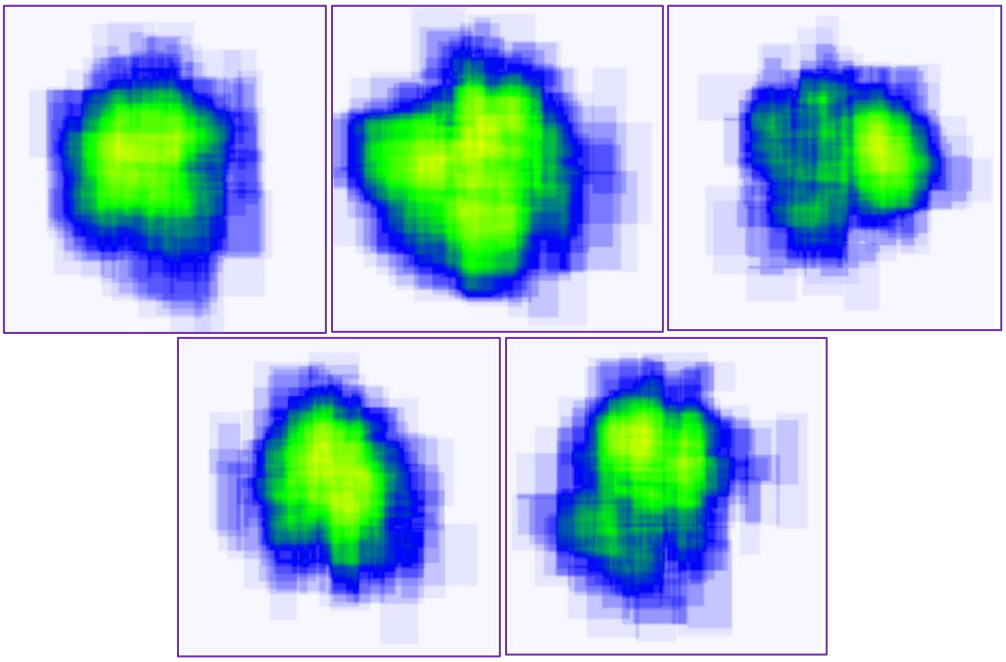


Figura 33- Mapa de Calor das Anotações

A maioria das imagens contém 5 blocos, com algumas imagens contendo até 7 blocos, Figura *34*.

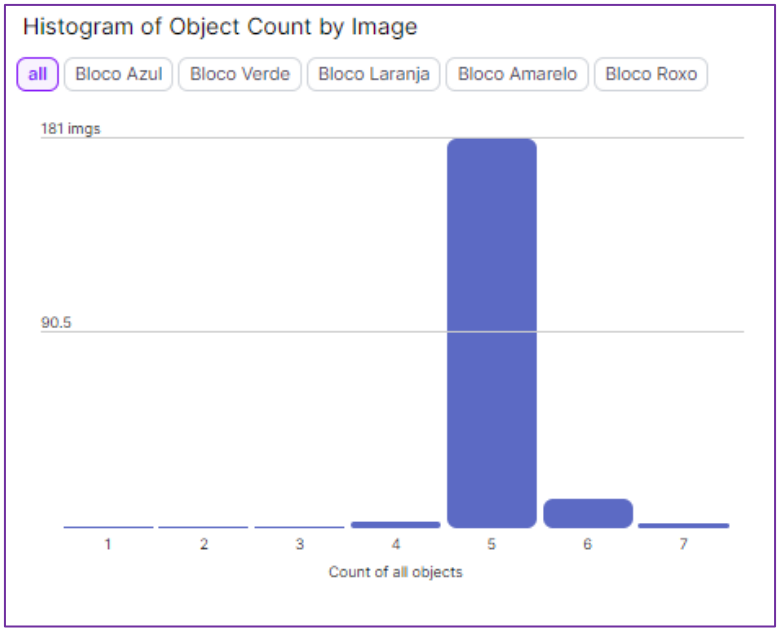


Figura 34- Histograma da Contagem de Objetos por Imagem

O modelo de identificação de blocos apresenta os seguintes resultados, como se verifica na Figura *35*:

mAP: 99.0%

Precisão: 97.9%

Recall: 96.4%

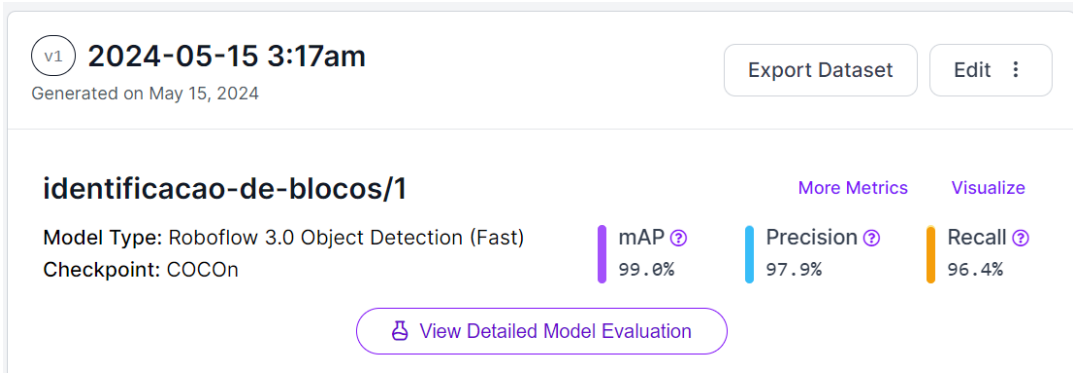


Figura 35- Avaliação do Modelo

Na Figura *36* é possível observar a precisão média durante a validação e durante o teste do modelo.

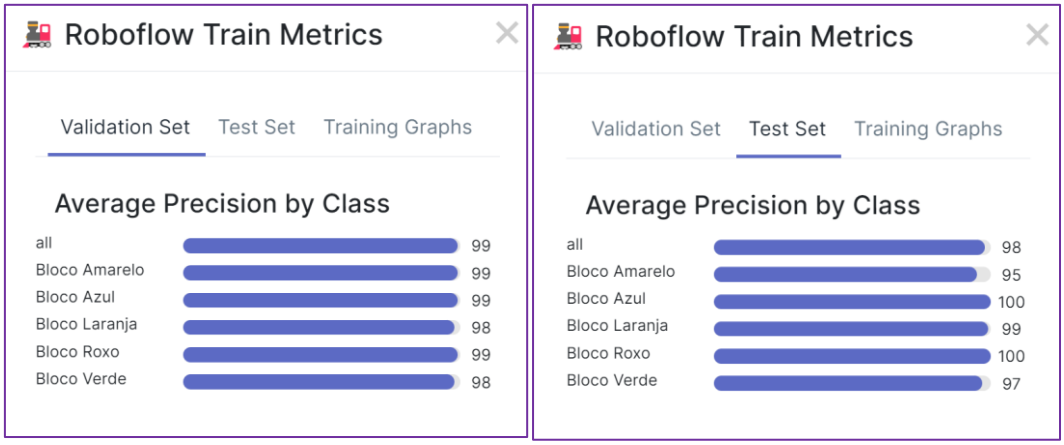


Figura 36- Precisão Média por Classe no Conjunto de Validação e Teste, respetivamente – Identificação de Blocos

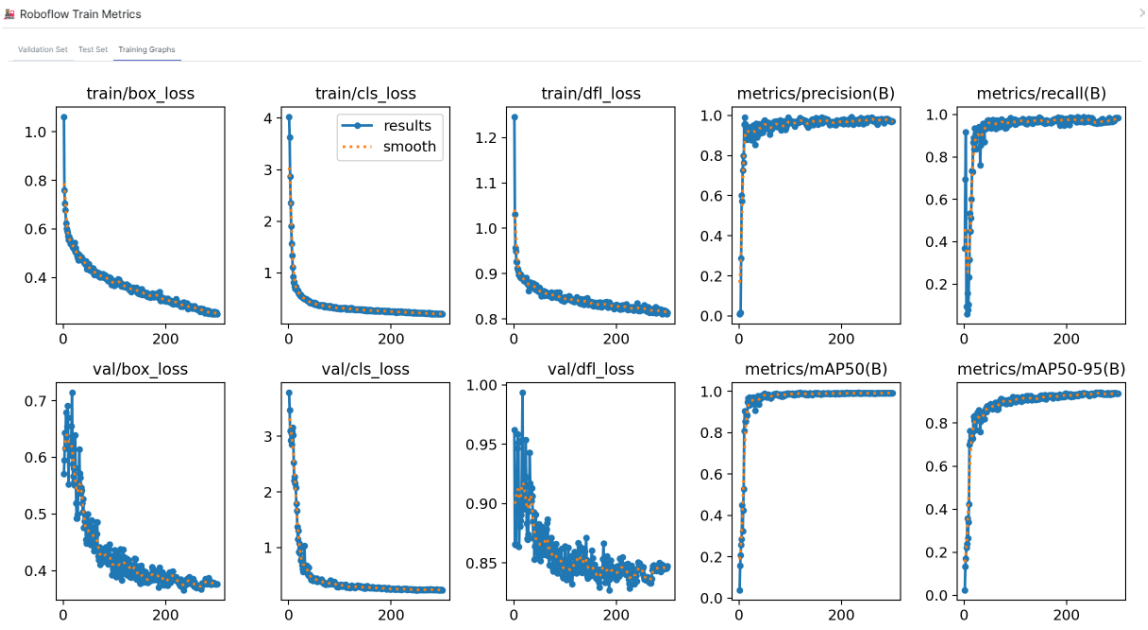


Figura 37- Métricas de Treinamento do Roboflow

As métricas apresentadas na Figura *37* fornecem uma visão detalhada do desempenho do modelo de deteção de *keypoints*. A métrica "train/box_loss" refere-se à perda na previsão das caixas delimitadoras durante o treino. A "train/cls_loss" mede a perda de classificação dos objetos dentro das caixas delimitadoras. A "train/dfl_loss" ajusta a perda com base na distribuição de confiança das previsões. A métrica de precisão (metrics/precision(B)) avalia a proporção de verdadeiros positivos em relação ao total de previsões, enquanto o recall (metrics/recall(B)) mede a capacidade do modelo de identificar todos os exemplos positivos. Ambas as curvas são altas, indicando alta exatidão e capacidade de identificação do modelo. A "val/box_loss" e a "val/cls_loss" medem as perdas nas previsões das caixas delimitadoras e na classificação nos dados de validação, mostrando que o modelo generaliza bem e mantém a precisão em dados não vistos. A "val/dfl_loss" confirma a melhoria na confiança das previsões em dados de validação. A "metrics/mAP50(B)" avalia a precisão média das previsões do modelo com um IoU (Intersection over Union) de 50%, indicando boa performance na deteção de objetos, enquanto a "metrics/mAP50-95(B)" avalia a precisão média em diferentes níveis de IoU, sugerindo que o modelo é robusto e preciso em diversos critérios de sobreposição.

4.1.2. Resultados da Deteção de Keypoints

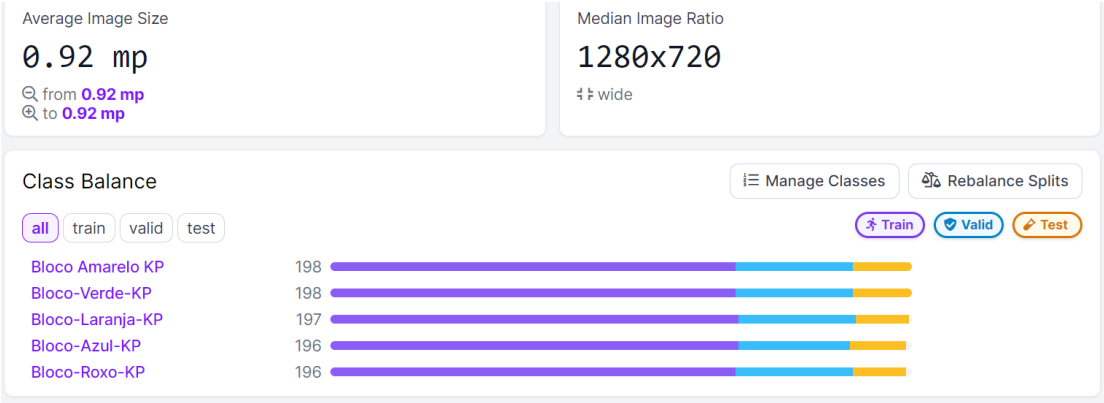


Figura 38- *Dataset Health Check* para *Keypoints*

O modelo de deteção de *keypoints* apresenta os seguintes resultados:

mAP: 98.3%

Precisão: 96.4%

Recall: 95.6%

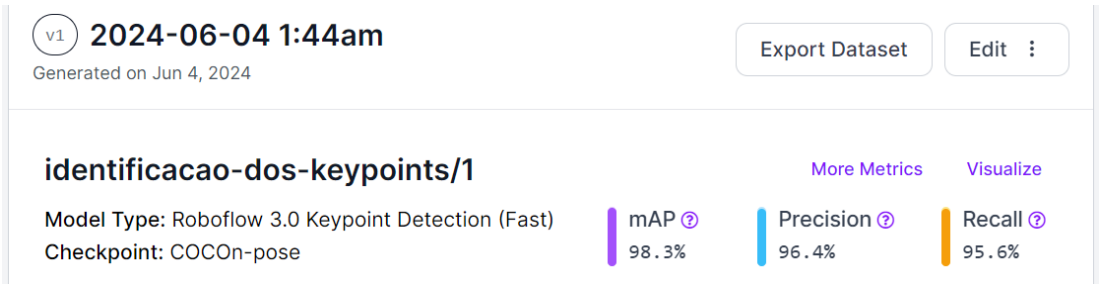


Figura 39- Avaliação do Modelo de Keypoints

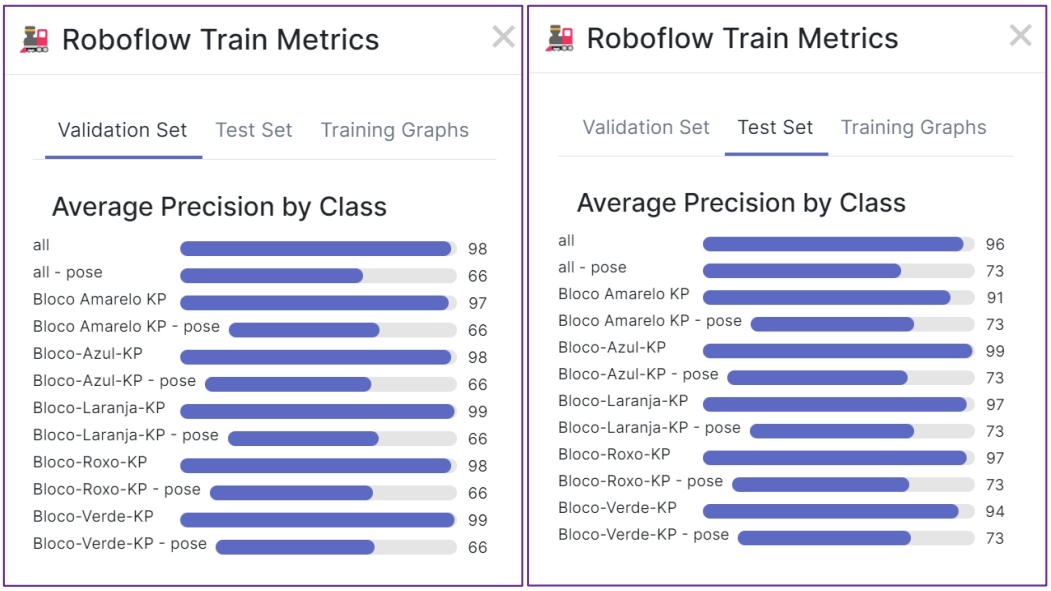


Figura 40- Precisão Média por Classe no Conjunto de Validação e Teste, respetivamente – Identificação dos *Keypoints*

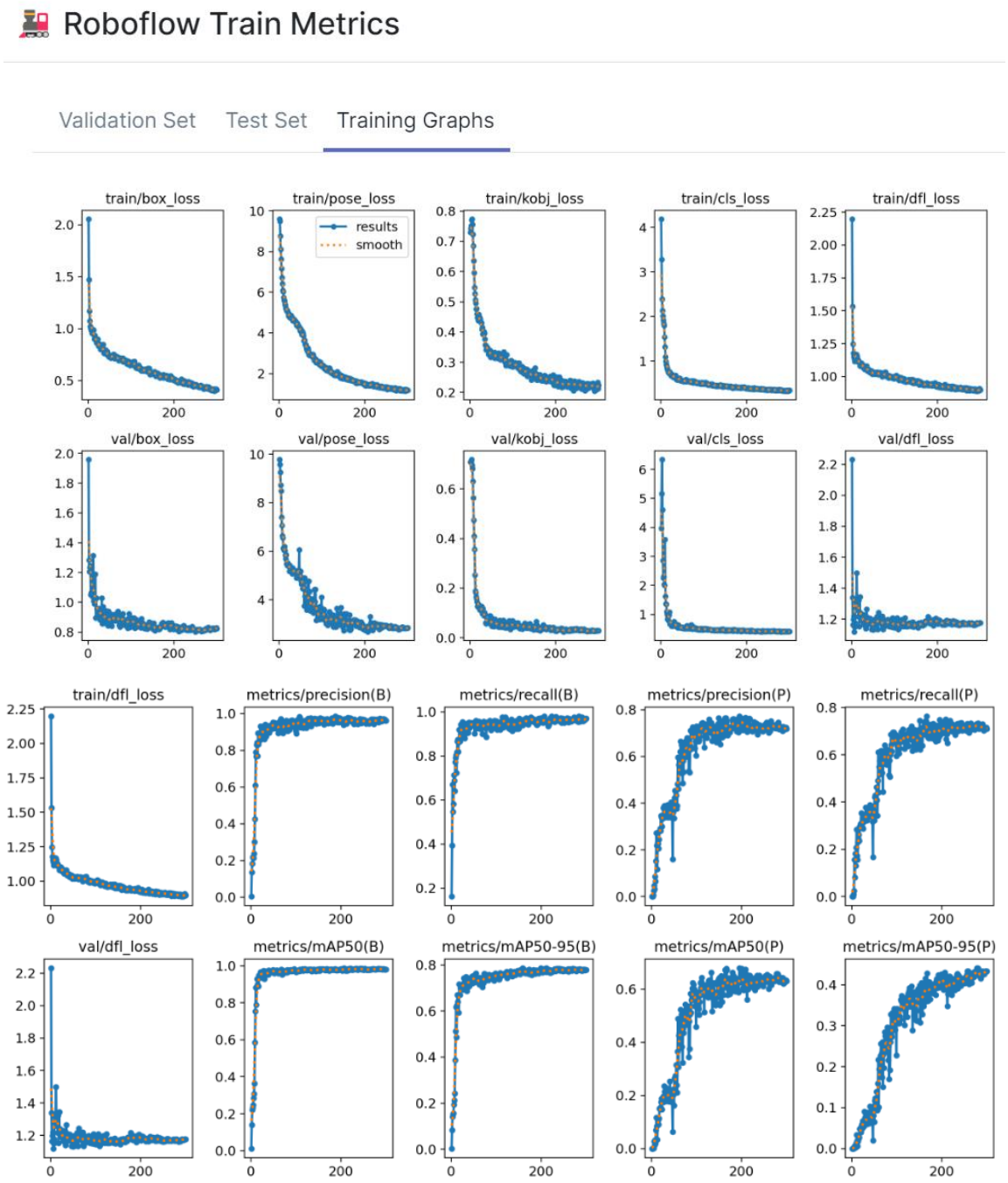


Figura 41- Métricas de Treinamento do Roboflow para Keypoints

As métricas apresentadas na Figura 41 fornecem uma visão detalhada do desempenho do modelo de deteção dos *keypoints*. A métrica "train/dfl_loss" refere-se à perda de "*distribution focal loss*" durante o treino, ajustando a perda com base na distribuição de confiança das previsões do modelo e penalizando previsões erradas de alta confiança. A curva desta métrica mostra uma diminuição constante, indicando que o modelo está a aprender a fazer predições mais confiáveis. As métricas de precisão (metrics/precision(B) e metrics/precision(P)) e recall (metrics/recall(B) e metrics/recall(P)) avaliam a exatidão e a capacidade do modelo de identificar corretamente *keypoints* e objetos. As curvas destas métricas mostram que o modelo

tem uma alta precisão e capacidade de identificação, com poucas predições incorretas. A "val/dfl_loss", similar à "train/dfl_loss", mas avaliada nos dados de validação. As métricas "metrics/mAP50(B)" e "metrics/mAP50-95(B)" avaliam a precisão média das previsões do modelo para *keypoints* a diferentes níveis de IoU, mostrando uma performance consistente e robusta. As métricas "metrics/mAP50(P)" e "metrics/mAP50-95(P)" fazem o mesmo para objetos, indicando eficiência na deteção. As curvas de perda (loss) decrescentes para os dados de treino e validação sugerem que o modelo está a melhorar a sua performance ao longo do tempo.

4.1.3. Precisão do Cálculo do Centro de Massa

Para avaliar a precisão do cálculo do centro de massa dos paralelepípedos, foram tiradas 50 fotos utilizando a câmara Intel RealSense D435i. As imagens foram processadas com o código desenvolvido para calcular o centro de massa dos objetos. Depois de recolhidos os dados, os centros de massa reais foram anotados manualmente, como mostra na Figura 42, e as distâncias entre os centros de massa calculados e reais foram medidas utilizando as coordenadas em *pixels*.

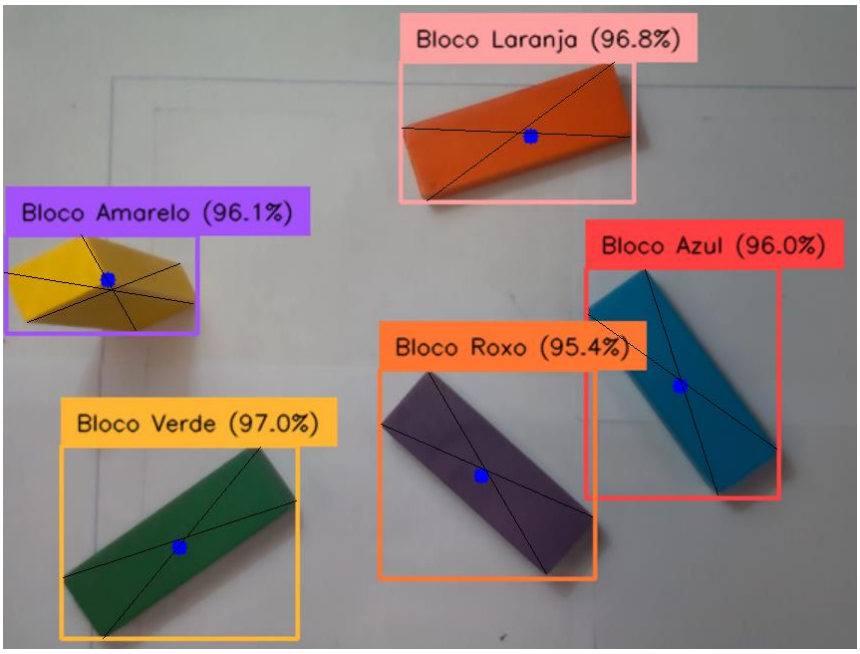


Figura 42- Anotação manual dos centro de massa

A Tabela 4 e a Figura 43 resumem as médias e os desvios padrão das principais métricas. Esta abordagem foi utilizada porque a distância diagonal representa uma medida padrão para a escala dos objetos, permitindo uma avaliação proporcional e relativa dos erros de deteção em diferentes tamanhos e posições dos blocos. Além disso, este método ajuda a manter a consistência da métrica de erro, independentemente da distância dos blocos em relação à câmara, facilitando a análise da precisão em um contexto mais amplo. O cálculo do centro de

massa dos paralelepípedos foi realizado utilizando uma folha de Excel, conforme ilustrado no Apêndice B.

Tabela 4- Erro de calculo do centro de massa por cor de bloco e desvio padrão

Bloco	Média	Desvio Padrão
Amarelo	2,50%	1,43%
Azul	2,61%	1,62%
Laranja	2,45%	1,29%
Roxo	3,28%	2,00%
Verde	2,34%	1,33%
Total	**2,64%**	**1,48%**

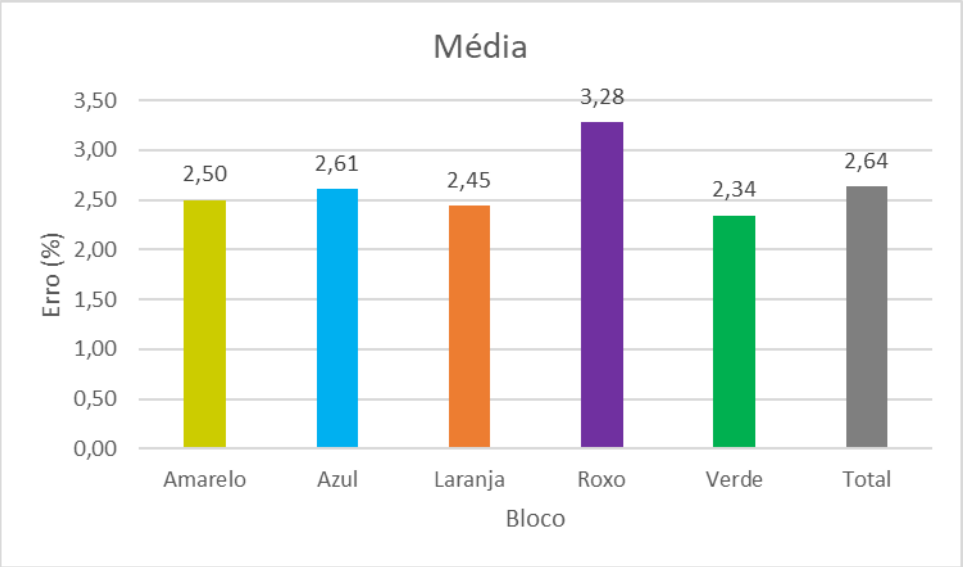


Figura 43- Erro percentual do calculo do centro de massa por bloco

4.2. Discussão de resultados

4.2.1. Desempenho dos Modelos

Os resultados indicam que o modelo de identificação de blocos e o modelo de deteção de *keypoints*, ambos treinados com o *Roboflow*, alcançaram níveis elevados de precisão e *recall*. No entanto, o modelo de identificação de Blocos mostrou uma precisão ligeiramente superior em comparação com o modelo de identificação dos *keypoints*. A seguir, detalhamos as possíveis razões para essas diferenças e a análise específica de cada modelo.

4.2.2. Comparação entre Modelos

Complexidade da Tarefa:

Identificação de Blocos: A tarefa de identificar objetos inteiros é relativamente mais simples. O modelo precisa detetar e classificar cada bloco como uma unidade completa, o que pode ser mais direto em termos de reconhecimento de padrões.

Identificação de *Keypoints*: Esta tarefa é intrinsecamente mais complexa, pois envolve a identificação de pontos específicos dentro de um objeto. Isso requer que o modelo tenha uma compreensão mais detalhada e precisa da estrutura interna do objeto, aumentando a dificuldade do treinamento e a variabilidade dos dados.

Ruído nos Dados:

Identificação de Blocos: As anotações para objetos inteiros tendem a ser mais consistentes, pois a localização e a forma do objeto são mais fáceis de definir e anotar.

Identificação dos *Keypoints*: As anotações de *keypoints* são mais propensas a erros e inconsistências devido à dificuldade de marcar pontos exatos. Pequenas variações nas anotações podem introduzir ruído significativo nos dados de treino, impactando negativamente a precisão do modelo.

Capacidade do Modelo e Requisitos Computacionais:

Identificação de Blocos: Os modelos utilizados para deteção de objetos geralmente requerem menos camadas e parâmetros em comparação com os modelos de *keypoints*, tornando o treinamento mais eficiente em termos de recursos computacionais.

Identificação dos *Keypoints*: Esses modelos precisam de redes neurais mais profundas e complexas para capturar os detalhes finos dos *keypoints*. Isso resulta em maior complexidade computacional, exigindo mais tempo de processamento e capacidade de hardware.

4.2.3. Análise do Bloco Verde

A análise do *heatmap* de anotações revelou que o bloco verde possui uma distribuição mais espalhada, indicando maior variabilidade na localização e forma do bloco verde nas imagens. Essa variabilidade pode ser um fator contribuinte para a menor precisão, pois:

Variabilidade na Localização: A maior dispersão dos *keypoints* para o bloco verde sugere que ele aparece em posições mais variadas nas imagens, dificultando para o modelo aprender um padrão consistente.

Impacto na Deteção: A variabilidade pode levar a dificuldades na deteção precisa dos *keypoints*, especialmente em condições de iluminação ou ângulos de visão diferentes, resultando em uma precisão geral mais baixa.

4.2.4. Comparação do Centro de Massa

Os resultados apresentados indicam variações na precisão do cálculo do centro de massa entre os diferentes blocos coloridos. A média do erro percentual para cada bloco, juntamente com o desvio padrão,

O bloco amarelo apresentou um erro percentual médio de 2,50%, indicando uma boa precisão. O desvio padrão de 1,43% sugere que os erros foram consistentes, com pouca variação entre as medições.

O bloco azul teve um erro médio ligeiramente superior, de 2,61%, com um desvio padrão de 1,62%, indicando uma variação maior nos erros, mas ainda dentro de uma faixa aceitável para aplicações práticas.

O bloco laranja mostrou o menor erro percentual médio, de 2,45%, indicando a maior precisão entre os blocos. O desvio padrão de 1,29% reflete uma alta consistência nas medições.

Em contraste, o bloco roxo apresentou o maior erro percentual médio, de 3,28%, e o maior desvio padrão, de 2,00%, sugerindo que a estimativa do centro de massa neste bloco foi a menos precisa e mais variável.

O bloco verde teve um erro médio de 2,34%, indicando uma boa precisão, com um desvio padrão de 1,33%, indicando uma variação relativamente baixa entre as medições.

A média total do erro percentual foi de 2,64%, com um desvio padrão de 1,48%. Esses valores refletem uma boa precisão geral do cálculo do centro de massa, com variações aceitáveis entre diferentes cores de blocos.

Os resultados mostram que o cálculo do centro de massa é geralmente preciso, com uma média de erro percentual que varia entre 2,34% e 3,28% para os diferentes blocos. A maior precisão foi observada para o bloco laranja, enquanto o bloco roxo apresentou a maior variação nos erros. Essas diferenças podem ser atribuídas a características específicas de cada cor e à forma como a luz e as sombras interagem com os blocos durante a captura das imagens. No geral, o desempenho do sistema é satisfatório para as aplicações logísticas.

5. Conclusão

Neste capítulo, apresentamos uma visão geral das principais conclusões obtidas a partir dos resultados e discussões dos capítulos anteriores. Também abordamos as limitações encontradas durante a pesquisa e sugerimos direções para trabalhos futuros.

5.1. Conclusões finais

Neste trabalho, foram desenvolvidos e treinados modelos de visão computacional para a identificação de blocos e de *keypoints*, utilizando técnicas avançadas de *deep learning* com a ferramenta YOLOv8. Os resultados obtidos demonstraram altos níveis de precisão e *recall* para ambos os modelos. Contudo, a tarefa de identificação de keypoints revelou-se mais complexa, em comparação com a identificação de blocos inteiros. A variabilidade nas anotações, especialmente para o bloco verde, foi um fator significativo que impactou a precisão do modelo de *keypoints*.

Os modelos de identificação de blocos e dos *keypoints*, quando executados individualmente, operam de maneira eficiente. No entanto, a combinação dos dois modelos num único código apresentou desafios significativos devido às limitações computacionais. Durante a execução combinada dos dois modelos, foram reveladas tremendas dificuldades devido elevada necessidade de poder computacional. A capacidade de processamento e a memória disponível foram insuficientes para lidar eficientemente com a carga combinada, resultando em tempos de execução prolongados e interrupções.

Um ponto de destaque é a precisão dos cálculos dos centros de massa. Os modelos conseguiram identificar os *keypoints* necessários para a determinação precisa dos centros de massa dos blocos, contribuindo para a eficácia das operações de manipulação e paletização automatizadas.

Outro aspeto inovador deste trabalho foi a utilização conjunta dos dois modelos de visão artificial. A integração de modelos de deteção de objetos e de *keypoints* num único sistema foi um avanço significativo, pois permitiu uma abordagem mais abrangente e detalhada na análise e manipulação dos blocos. Esta combinação inovadora de técnicas mostrou-se eficaz na identificação e manipulação precisa de objetos, destacando o potencial da visão computacional integrada para aplicações industriais.

5.2. Limitações e trabalhos futuros

Embora os resultados sejam encorajadores, esta pesquisa enfrentou várias limitações que podem ser abordadas em trabalhos futuros. Em primeiro lugar, a integração dos dois modelos num único sistema revelou limitações computacionais significativas. Técnicas de otimização de recursos ou a adoção de *hardware* mais avançado, sendo que o *hardware* utilizado conta com as seguintes características: Processador: Intel(R) Core(TM) i5-10210U CPU; Velocidade do Processador:2.1GHz; Memória (RAM):16GB.

A variabilidade nas anotações de *keypoints*, particularmente no caso do bloco verde, sugere a necessidade de aprimoramentos nos processos de anotação para reduzir inconsistências. Trabalhos futuros podem se concentrar em melhorar esses processos, possivelmente através de técnicas automatizadas de anotação.

Adicionalmente, os modelos foram treinados e avaliados com um conjunto de dados específico. Estudos futuros devem investigar a generalização desses modelos para diferentes conjuntos de dados e contextos de aplicação, visando validar sua robustez e flexibilidade.

Um próximo passo importante seria programar o sistema para detetar a orientação da face dos blocos. Esta funcionalidade é crucial para preparar a aplicação dos modelos em um braço robótico, possibilitando a manipulação precisa e eficiente dos blocos em ambientes industriais.

Referências

[1] S. Y. Nof, *Handbook of industrial robotics*. John Wiley & Sons, 1999.

[2] C. Heyer, "Human-robot interaction and future industrial robotics applications," in *2010 ieee/rsj international conference on intelligent robots and systems*, IEEE, 2010, pp. 4749–4754.

[3] V. F. Romano and M. Dutra, "Introdução a robótica industrial," *Robótica Ind. Apl. na Indústria Manufatura e Process. São Paulo Edgard Blücher*, pp. 1–19, 2002.

[4] F. A. de S. Pereira and J. Machado, *Robótica Industrial Parte I - Introdução, Programação Básica e Manutenção*. Porto: Quântica Editora, 2023.

[5] V. M. F. Santos, "Robótica industrial," *Univ. Aveito-Departamento Eng. Mecânica*, 2004.

[6] T. Pham, "Position control of cartesian robot," 2019.

[7] R. S. Mole, A. M. Rode, K. N. Phadtare, P. D. Patil, and J. B. Satpute, "A Literature Review on Structural Properties of Different Types of Robots".

[8] A. A. Santos, F. Pereira, and C. Felgueiras, "Optimization and improving of the production capacity of a flexible tyre painting cell," *Int. J. Adv. Manuf. Technol.*, pp. 1–13, 2024.

[9] K. Ono, T. Hayashi, M. Fujii, N. Shibasaki, and M. Sonehara, "Development for industrial robotics applications," *IHI Eng. Rev.*, vol. 42, no. 2, pp. 103–107, 2009.

[10] P. Kah, M. Shrestha, E. Hiltunen, and J. Martikainen, "Robotic arc welding sensors and programming in industrial applications," *Int. J. Mech. Mater. Eng.*, vol. 10, no. 1, pp. 1–16, 2015.

[11] R. Bloss, "Collaborative robots are rapidly providing major improvements in productivity, safety, programing ease, portability and cost while addressing many new applications," *Ind. Robot An Int. J.*, vol. 43, no. 5, pp. 463–468, 2016.

[12] Å. Fast-Berglund, F. Palmkvist, P. Nyqvist, S. Ekered, and M. Åkerman, "Evaluating cobots for final assembly," *Procedia CIRP*, vol. 44, pp. 175–180, 2016.

[13] M. Knudsen and J. Kaivo-Oja, "Collaborative robots: Frontiers of current literature," *J. Intell. Syst. Theory Appl.*, vol. 3, no. 2, pp. 13–20, 2020.

[14] S. El Zaatari, M. Marei, W. Li, and Z. Usman, "Cobot programming for collaborative industrial tasks: An overview," *Rob. Auton. Syst.*, vol. 116, pp. 162–180, 2019.

[15] F. Platbrood and O. Gornemann, "Safe robotics-safety in collaborative robot systems," *Sick AG, Waldkirch, Ger.*, vol. 980, 2017.

[16] A. A. Santos, J. Haladus, F. Pereira, C. Felgueiras, and R. Fazenda, "Simulation Case Study for Improving Painting Tires Process Using the Fanuc Roboguide Software," in *International Conference on Flexible Automation and Intelligent Manufacturing*, Springer, 2023, pp. 517–524.

[17] F. Sherwani, M. M. Asad, and B. S. K. K. Ibrahim, "Collaborative robots and industrial revolution 4.0 (ir 4.0)," in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, IEEE, 2020, pp. 1–5.

[18] M. Fujii, H. Murakami, and M. Sonehara, “Study on application of a human-robot collaborative system using hand-guiding in a production line,” *IHI Eng. Rev*, vol. 49, no. 1, pp. 24–29, 2016.

[19] T. Saito, T. Hoshi, H. Ikeda, and K. Okabe, “Global harmonization of safety regulations for the use of industrial robots-permission of collaborative operation and a related study by JNIOSH,” *Ind. Health*, vol. 53, no. 6, pp. 498–504, 2015.

[20] L. Pérez, Í. Rodríguez, N. Rodríguez, R. Usamentiaga, and D. F. García, “Robot guidance using machine vision techniques in industrial environments: A comparative review,” *Sensors*, vol. 16, no. 3, p. 335, 2016.

[21] R. Usamentiaga, J. Molleda, D. F. Garcia, L. Pérez, and G. Vecino, “Real-time line scan extraction from infrared images using the wedge method in industrial environments,” *J. Electron. Imaging*, vol. 19, no. 4, p. 43017, 2010.

[22] B. R. Barbero and E. S. Ureta, “Comparative study of different digitization techniques and their accuracy,” *Comput. Des.*, vol. 43, no. 2, pp. 188–206, 2011.

[23] J. Pages, C. Collewet, F. Chaumette, and J. Salvi, “A camera-projector system for robot positioning by visual servoing,” in *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW’06)*, IEEE, 2006, p. 2.

[24] J. Salvi, *An approach to coded structured light to obtain three dimensional information*. Universitat de Girona, 1998.

[25] M. Viager, “Analysis of Kinect for mobile robots,” *Lyngby Tech. Univ. Denmark*, 2011.

[26] K. Khoshelham and S. O. Elberink, “Accuracy and resolution of kinect depth data for indoor mapping applications,” *sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[27] M. Mahmud, D. Joannic, M. Roy, A. Isheil, and J.-F. Fontaine, “3D part inspection path planning of a laser scanner with control on the uncertainty,” *Comput. Des.*, vol. 43, no. 4, pp. 345–355, 2011.

[28] J. Rutinowski, H. Youssef, A. Gouda, C. Reining, and M. Roidl, “The potential of deep learning based computer vision in warehousing logistics,” *Logist. J. Proc.*, vol. 2022, no. 18, 2022.

[29] M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui, and J. Collomosse, “Everything you wanted to know about deep learning for computer vision but were afraid to ask,” in *2017 30th SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)*, IEEE, 2017, pp. 17–41.

[30] N. Correll *et al.*, “Analysis and observations from the first amazon picking challenge,” *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 172–188, 2016.

[31] S. Surati, S. Hedaoo, T. Rotti, V. Ahuja, and N. Patel, “Pick and place robotic arm: a review paper,” *Int. Res. J. Eng. Technol*, vol. 8, no. 2, pp. 2121–2129, 2021.

[32] N. Sobhan and A. S. Shaikat, “Implementation of Pick & Place Robotic Arm for Warehouse Products Management,” in *2021 IEEE 7th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, IEEE, 2021, pp. 156–161.

[33] J. L. Arévalo-Hernández, E. Rubio-Espino, V. H. Ponce-Ponce, and J. H. Sossa-Azuela, “Vision assisted pick and place robotic machine.,” *Int. J. Comb. Optim. Probl. Informatics*, vol. 12, no. 3, 2021.

[34] H.-I. Lin, Y.-Y. Chen, and Y.-Y. Chen, “Robot vision to recognize both object and rotation for robot pick-and-place operation,” in *2015 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*, IEEE, 2015, pp. 1–6.

[35] S. Sultonov, “IMPORTANCE OF PYTHON PROGRAMMING LANGUAGE IN MACHINE LEARNING.,” *Int. Bull. Eng. Technol.*, vol. 3, no. 9, pp. 28–30, 2023.

[36] Z. DeVito, J. Ansel, W. Constable, M. Suo, A. Zhang, and K. Hazelwood, “Using python for model inference in deep learning,” *arXiv Prepr. arXiv2104.00254*, 2021.

[37] S. Alexandrova, Z. Tatlock, and M. Cakmak, “RoboFlow: A flow-based visual programming language for mobile manipulation tasks,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 5537–5544.

[38] Q. Lin, G. Ye, J. Wang, and H. Liu, “Roboflow: a data-centric workflow management system for developing ai-enhanced robots,” in *Conference on Robot Learning*, PMLR, 2022, pp. 1789–1794.

[39] M. Hussain, “YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection,” *Machines*, vol. 11, no. 7, p. 677, 2023.

[40] P. Potrimba, “What is Keypoint Detection?,” *Roboflow Blog*, 2023, [Online]. Available: https://blog.roboflow.com/what-is-keypoint-detection/

[41] J. Nelson, “Train Test Split Guide and Overview,” *Roboflow Blog*, Oct. 2020, [Online]. Available: https://blog.roboflow.com/train-test-split-with-roboflow/

[42] M. Walia, “Overfitting in Machine Learning and Computer Vision,” *Roboflow Blog*. [Online]. Available: https://blog.roboflow.com/overfitting-machine-learning-computer-vision/

[43] P. Potrimba, “What is Hyperparameter Tuning? A Deep Dive.,” *Roboflow Blog*. [Online]. Available: https://blog.roboflow.com/what-is-hyperparameter-tuning/

Declaração de Integridade

Declaro ter conduzido este trabalho académico com integridade. Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração.

Declaro que o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim.

Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P.PORTO.

NOME: Miguel Delgado Silveira

ISEP, Porto, 14 de junho de 2024

Apêndice A

Este guia fornece instruções detalhadas para executar o programa desenvolvido para a identificação de objetos e *keypoints*.

Pré-requisitos

Antes de iniciar a execução do programa, certifique-se de que os seguintes pré-requisitos estão atendidos:

Instalação do Python: O programa Python deve estar instalado no seu sistema. Recomenda-se a versão 3.7 ou superior.

Bibliotecas Necessárias: As seguintes bibliotecas Python devem ser instaladas:

numpy

opencv-python

roboflow

supervision

inference

Passo a Passo para Execução

Utilizar o Código Fornecido no Relatório

Copie o código fornecido no relatório para arquivos Python no diretório local. O código pode ser encontrado nas seções "Código de Identificação de Blocos", "Código de Identificação dos Keypoints" e “Código de Integração dos Modelos” do relatório.

Configurar a API Key do Roboflow

Obtenha a API Key do Roboflow. Esta chave é necessária para ter acesso aos *datasets* e modelos treinados.

Defina a API Key no ambiente de execução:

set ROBOFLOW_API_KEY=<API_KEY>

Executar o Código de Identificação de Blocos

Na Linha de Comandos executar o *Script*:

python identificacao_blocos.py

O código capturará *frames* de vídeo da câmara (referida por video_reference), realizará a inferência para detetar os paralelepípedos e exibirá as imagens anotadas com as caixas delimitadoras ao redor dos objetos detetados.

Executar o Código de Identificação dos *Keypoints*

Da mesma forma, execute o script de identificação dos *keypoints*:

python identificacao_keypoints.py

Este código identificará os vértices dos paralelepípedos nas imagens capturadas e calculará o centro de massa de cada bloco, exibindo as imagens anotadas com os *keypoints* e o centro de massa.

Executar o Código que Usa os Dois Modelos

Para executar o código que integra ambos os modelos, copie o código fornecido no relatório para um arquivo Python (por exemplo, integracao_modelos.py) e execute-o:

python integracao_modelos.py

Este script combinará a deteção de blocos e a identificação de keypoints, realizando ambas as tarefas em paralelo. O código capturará frames de vídeo, executará a inferência utilizando ambos os modelos, e exibirá as imagens anotadas com as caixas delimitadoras e os keypoints identificados.

Apêndice B

Imagem		x Calculado	y Calculado	x Real	y Real	Distância (Pixeis)	Erro	Erro (%)
1	Amarelo	752	299	754	295	4,472135955	0,018997	1,89972
	Azul	713	497	720	499	7,280109889	0,030925	3,09252
	Laranja	555	518	550	510	9,433981132	0,040075	4,007464
	Roxo	324	427	323	423	4,123105626	0,017515	1,751455
	Verde							
2	Amarelo	780	306	781	303	3,16227766	0,013433	1,343305
	Azul	688	506	692	505	4,123105626	0,017515	1,751455
	Laranja	550	462	547	457	5,830951895	0,024769	2,476932
	Roxo	313	441	319	434	9,219544457	0,039164	3,916373
	Verde							
3	Amarelo	743	336	740	329	7,615773106	0,032351	3,235106
	Azul	628	500	627	497	3,16227766	0,013433	1,343305
	Laranja							
	Roxo	376	577	381	570	8,602325267	0,036542	3,654184
	Verde							
4	Amarelo	743	329	740	325	5	0,02124	2,123951
	Azul	629	497	625	495	4,472135955	0,018997	1,89972
	Laranja	360	417	360	409	8	0,033983	3,398322
	Roxo	354	616	354	627	11	0,046727	4,672693
	Verde	459	539	458	545	6,08276253	0,025839	2,583898
5	Amarelo							
	Azul	760	467	765	466	5,099019514	0,02166	2,166014
	Laranja	379	357	377	355	2,828427125	0,012015	1,201488
	Roxo	474	585	484	583	10,19803903	0,04332	4,332028
	Verde	486	99	485	100	1,414213562	0,006007	0,600744
6	Amarelo	783	218	787	217	4,123105626	0,017515	1,751455
	Azul	760	468	765	467	5,099019514	0,02166	2,166014
	Laranja	241	390	241	384	6	0,025487	2,548742
	Roxo	478	584	478	584			
	Verde	498	158	508	153	11,18033989	0,047493	4,7493
7	Amarelo	777	207	779	212	5,385164807	0,022876	2,287566
	Azul	757	465	764	465	7	0,029735	2,973532
	Laranja	322	354	324	351	3,605551275	0,015316	1,531603
	Roxo	472	584	480	581	8,544003745	0,036294	3,62941
	Verde	490	140	488	139	2,236067977	0,009499	0,94986
8	Amarelo	733	214	738	215	5,099019514	0,02166	2,166014
	Azul	752	484	754	484	2	0,008496	0,849581
	Laranja	320	358	324	353	6,403124237	0,0272	2,719985
	Roxo	470	584	477	583	7,071067812	0,030037	3,003721
	Verde	484	143	481	144	3,16227766	0,013433	1,343305
9	Amarelo	731	196	736	196	5	0,02124	2,123951
	Azul	752	465	749	462	4,242640687	0,018022	1,802233
	Laranja	321	355	323	352	3,605551275	0,015316	1,531603
	Roxo	467	582	450	582	17	0,072214	7,221435
	Verde	484	142	481	147	5,830951895	0,024769	2,476932
10	Amarelo	666	205	659	200	8,602325267	0,036542	3,654184
	Azul	756	464	752	464	4	0,016992	1,699161
	Laranja	262	373	266	378	6,403124237	0,0272	2,719985
	Roxo	387	133	384	130	4,242640687	0,018022	1,802233
	Verde	525	397	524	394	3,16227766	0,013433	1,343305

11	Amarelo	596	452	598	447	5,385164807	0,022876	2,287566
	Azul	352	449	354	453	4,472135955	0,018997	1,89972
	Laranja	177	336	183	334	6,32455532	0,026866	2,68661
	Roxo							
	Verde	327	107	327	110	3	0,012744	1,274371
12	Amarelo	751	298	752	296	2,236067977	0,009499	0,94986
	Azul	586	465	566	463	20,09975124	0,085382	8,538179
	Laranja	276	398	280	396	4,472135955	0,018997	1,89972
	Roxo	609	103	602	107	8,062257748	0,034248	3,424769
	Verde	316	165	322	165	6	0,025487	2,548742
13	Amarelo	766	264	764	264	2	0,008496	0,849581
	Azul	598	469	595	472	4,242640687	0,018022	1,802233
	Laranja	304	413	306	408	5,385164807	0,022876	2,287566
	Roxo	571	129	575	131	4,472135955	0,018997	1,89972
	Verde	344	165	348	167	4,472135955	0,018997	1,89972
14	Amarelo	740	360	753	363	13,34166406	0,056674	5,667409
	Azul	523	569	526	580	11,40175425	0,048434	4,843354
	Laranja	295	282	297	281	2,236067977	0,009499	0,94986
	Roxo	536	308	533	301	7,615773106	0,032351	3,235106
	Verde	568	98	571	101	4,242640687	0,018022	1,802233
15	Amarelo	723	446	721	440	6,32455532	0,026866	2,68661
	Azul							
	Laranja	231	329	234	328	3,16227766	0,013433	1,343305
	Roxo							
	Verde	599	233	599	228	5	0,02124	2,123951
16	Amarelo	723	450	724	442	8,062257748	0,034248	3,424769
	Azul	492	601	491	603	2,236067977	0,009499	0,94986
	Laranja	253	235	257	235	4	0,016992	1,699161
	Roxo	249	516	255	541	25,70992026	0,109213	10,92132
	Verde							
17	Amarelo	667	436	659	432	8,94427191	0,037994	3,79944
	Azul	421	595	426	595	5	0,02124	2,123951
	Laranja	230	153	234	149	5,656854249	0,02403	2,402977
	Roxo	113	441	114	436	5,099019514	0,02166	2,166014
	Verde	599	118	597	118	2	0,008496	0,849581
18	Amarelo	574	364	574	357	7	0,029735	2,973532
	Azul	303	554	300	546	8,544003745	0,036294	3,62941
	Laranja	205	151	205	149	2	0,008496	0,849581
	Roxo	151	339	159	334	9,433981132	0,040075	4,007464
	Verde	588	132	585	130	3,605551275	0,015316	1,531603
19	Amarelo	578	367	580	359	8,246211251	0,035029	3,50291
	Azul	308	557	304	552	6,403124237	0,0272	2,719985
	Laranja	235	66	239	71	6,403124237	0,0272	2,719985
	Roxo	157	341	161	336	6,403124237	0,0272	2,719985
	Verde	592	136	585	131	8,602325267	0,036542	3,654184
20	Amarelo	825	177	827	166	11,18033989	0,047493	4,7493
	Azul	394	456	392	455	2,236067977	0,009499	0,94986
	Laranja	330	182	334	182	4	0,016992	1,699161
	Roxo	636	475	626	469	11,66190379	0,049539	4,953863
	Verde	606	173	610	166	8,062257748	0,034248	3,424769

21	Amarelo	825	177	827	166	11,18033989	0,047493	4,7493
	Azul	394	456	392	455	2,236067977	0,009499	0,94986
	Laranja	330	182	334	182	4	0,016992	1,699161
	Roxo	636	475	625	468	13,03840481	0,055386	5,538588
	Verde	606	173	609	166	7,615773106	0,032351	3,235106
22	Amarelo	853	166	853	162	4	0,016992	1,699161
	Azul	482	508	483	500	8,062257748	0,034248	3,424769
	Laranja	337	87	343	86	6,08276253	0,025839	2,583898
	Roxo	641	270	640	266	4,123105626	0,017515	1,751455
	Verde	277	299	277	297	2	0,008496	0,849581
23	Amarelo							
	Azul	490	536	488	526	10,19803903	0,04332	4,332028
	Laranja	596	312	594	305	7,280109889	0,030925	3,09252
	Roxo	779	407	773	401	8,485281374	0,036045	3,604465
	Verde	311	334	315	330	5,656854249	0,02403	2,402977
24	Amarelo							
	Azul	477	554	477	547	7	0,029735	2,973532
	Laranja	563	187	561	183	4,472135955	0,018997	1,89972
	Roxo	809	427	809	420	7	0,029735	2,973532
	Verde	380	347	383	348	3,16227766	0,013433	1,343305
25	Amarelo	355	144	361	145	6,08276253	0,025839	2,583898
	Azul	571	560	573	560	2	0,008496	0,849581
	Laranja	547	369	540	367	7,280109889	0,030925	3,09252
	Roxo	175	543	182	536	9,899494937	0,042052	4,205209
	Verde	248	343	251	344	3,16227766	0,013433	1,343305
26	Amarelo	583	143	578	143	5	0,02124	2,123951
	Azul	379	600	388	593	11,40175425	0,048434	4,843354
	Laranja	561	362	550	359	11,40175425	0,048434	4,843354
	Roxo	294	182	295	184	2,236067977	0,009499	0,94986
	Verde	278	371	276	370	2,236067977	0,009499	0,94986
27	Amarelo	518	180	518	178	2	0,008496	0,849581
	Azul	385	530	383	532	2,828427125	0,012015	1,201488
	Laranja	653	415	646	408	9,899494937	0,042052	4,205209
	Roxo	302	187	297	186	5,099019514	0,02166	2,166014
	Verde	181	367	188	368	7,071067812	0,030037	3,003721
28	Amarelo	557	211	557	211	0	0	0
	Azul	395	494	395	494	0	0	0
	Laranja	593	366	593	366	0	0	0
	Roxo					0		0
	Verde	178	359	182	360	4,123105626	0,017515	1,751455
29	Amarelo							
	Azul	408	586	409	576	10,04987562	0,042691	4,26909
	Laranja	643	341	641	339	2,828427125	0,012015	1,201488
	Roxo	261	202	271	212	14,14213562	0,060074	6,007442
	Verde	131	427	135	428	4,123105626	0,017515	1,751455
30	Amarelo	513	110	521	113	8,544003745	0,036294	3,62941
	Azul	405	587	409	574	13,60147051	0,057778	5,777773
	Laranja	624	374	624	368	6	0,025487	2,548742
	Roxo	243	206	250	203	7,615773106	0,032351	3,235106
	Verde	180	477	182	473	4,472135955	0,018997	1,89972

31	Amarelo	447	281	447	276	5	0,02124	2,123951
	Azul	465	534	465	530	4	0,016992	1,699161
	Laranja							
	Roxo	244	260	256	255	13	0,055223	5,522274
	Verde	166	464	173	459	8,602325267	0,036542	3,654184
32	Amarelo	408	211	409	209	2,236067977	0,009499	0,94986
	Azul	459	489	461	491	2,828427125	0,012015	1,201488
	Laranja	547	320	553	317	6,708203932	0,028496	2,84958
	Roxo	204	217	208	220	5	0,02124	2,123951
	Verde	211	492	214	485	7,615773106	0,032351	3,235106
33	Amarelo	271	334	279	336	8,246211251	0,035029	3,50291
	Azul	578	339	576	342	3,605551275	0,015316	1,531603
	Laranja	483	144	483	142	2	0,008496	0,849581
	Roxo	166	517	171	517	5	0,02124	2,123951
	Verde	501	556	499	550	6,32455532	0,026866	2,68661
34	Amarelo	189	177	183	177	6	0,025487	2,548742
	Azul	621	342	625	342	4	0,016992	1,699161
	Laranja	488	159	485	157	3,605551275	0,015316	1,531603
	Roxo	144	473	154	463	14,14213562	0,060074	6,007442
	Verde	501	566	504	563	4,242640687	0,018022	1,802233
35	Amarelo	229	206	227	201	5,385164807	0,022876	2,287566
	Azul	646	361	643	361	3	0,012744	1,274371
	Laranja	513	146	510	148	3,605551275	0,015316	1,531603
	Roxo	127	466	128	461	5,099019514	0,02166	2,166014
	Verde	477	595	473	588	8,062257748	0,034248	3,424769
36	Amarelo	269	274	267	272	2,828427125	0,012015	1,201488
	Azul	699	339	691	334	9,433981132	0,040075	4,007464
	Laranja	483	165	487	175	10,77032961	0,045751	4,575131
	Roxo	168	509	168	505	4	0,016992	1,699161
	Verde	503	548	497	539	10,81665383	0,045948	4,594809
37	Amarelo	327	215	328	213	2,236067977	0,009499	0,94986
	Azul							
	Laranja	498	131	504	131	6	0,025487	2,548742
	Roxo	203	531	200	526	5,830951895	0,024769	2,476932
	Verde	540	544	537	540	5	0,02124	2,123951
38	Amarelo	245	281	249	281	4	0,016992	1,699161
	Azul	680	346	689	340	10,81665383	0,045948	4,594809
	Laranja	508	173	509	166	7,071067812	0,030037	3,003721
	Roxo	194	574	190	570	5,656854249	0,02403	2,402977
	Verde	505	559	508	561	3,605551275	0,015316	1,531603
39	Amarelo	216	297	219	293	5	0,02124	2,123951
	Azul	680	336	685	337	5,099019514	0,02166	2,166014
	Laranja	449	189	456	183	9,219544457	0,039164	3,916373
	Roxo	191	570	189	567	3,605551275	0,015316	1,531603
	Verde	509	526	509	523	3	0,012744	1,274371
40	Amarelo	269	301	274	303	5,385164807	0,022876	2,287566
	Azul	696	381	705	375	10,81665383	0,045948	4,594809
	Laranja							
	Roxo	560	522	559	520	2,236067977	0,009499	0,94986
	Verde	137	546	126	549	11,40175425	0,048434	4,843354

41	Amarelo	105	326	110	336	11,18033989	0,047493	4,7493
	Azul	679	432	680	427	5,099019514	0,02166	2,166014
	Laranja	529	181	517	178	12,36931688	0,052544	5,254366
	Roxo	480	522	484	516	7,211102551	0,030632	3,063206
	Verde	177	594	184	583	13,03840481	0,055386	5,538588
42	Amarelo	537	158	535	158	2	0,008496	0,849581
	Azul	675	425	677	423	2,828427125	0,012015	1,201488
	Laranja	153	308	148	302	7,810249676	0,033177	3,317718
	Roxo	432	460	443	460	11	0,046727	4,672693
	Verde	176	583	179	579	5	0,02124	2,123951
43	Amarelo	647	147	644	146	3,16227766	0,013433	1,343305
	Azul	671	452	669	448	4,472135955	0,018997	1,89972
	Laranja							
	Roxo	422	317	424	314	3,605551275	0,015316	1,531603
	Verde	200	591	207	582	11,40175425	0,048434	4,843354
44	Amarelo	619	223	617	224	2,236067977	0,009499	0,94986
	Azul	560	532	567	529	7,615773106	0,032351	3,235106
	Laranja	122	291	124	286	5,385164807	0,022876	2,287566
	Roxo	384	294	381	288	6,708203932	0,028496	2,84958
	Verde	277	550	278	546	4,123105626	0,017515	1,751455
45	Amarelo	649	292	636	294	13,15294644	0,055872	5,587244
	Azul	507	570	512	565	7,071067812	0,030037	3,003721
	Laranja	121	259	124	261	3,605551275	0,015316	1,531603
	Roxo	337	267	342	265	5,385164807	0,022876	2,287566
	Verde	124	561	128	561	4	0,016992	1,699161
46	Amarelo	648	359	644	357	4,472135955	0,018997	1,89972
	Azul	514	576	516	572	4,472135955	0,018997	1,89972
	Laranja	297	609	296	604	5,099019514	0,02166	2,166014
	Roxo	349	293	354	288	7,071067812	0,030037	3,003721
	Verde	142	414	146	415	4,123105626	0,017515	1,751455
47	Amarelo	566	348	563	339	9,486832981	0,040299	4,029914
	Azul	537	586	526	586	11	0,046727	4,672693
	Laranja							
	Roxo	383	292	378	293	5,099019514	0,02166	2,166014
	Verde	120	460	122	454	6,32455532	0,026866	2,68661
48	Amarelo	632	307	624	306	8,062257748	0,034248	3,424769
	Azul	507	595	502	589	7,810249676	0,033177	3,317718
	Laranja	245	647	251	646	6,08276253	0,025839	2,583898
	Roxo							
	Verde	98	432	99	433	1,414213562	0,006007	0,600744
49	Amarelo	625	275	620	275	5	0,02124	2,123951
	Azul	511	587	505	591	7,211102551	0,030632	3,063206
	Laranja	282	604	282	599	5	0,02124	2,123951
	Roxo	217	400	223	401	6,08276253	0,025839	2,583898
	Verde	270	149	276	154	7,810249676	0,033177	3,317718
50	Amarelo	786	328	780	323	7,810249676	0,033177	3,317718
	Azul	616	578	615	573	5,099019514	0,02166	2,166014
	Laranja	397	129	401	138	9,848857802	0,041837	4,183699
	Roxo	140	487	139	478	9,055385138	0,038466	3,84664
	Verde	398	457	400	451	6,32455532	0,026866	2,68661

Apêndice C

```python
from inference import InferencePipeline
from inference.core.interfaces.camera.entities import VideoFrame

# import opencv to display our annotated images
import cv2
# import supervision to help visualize our predictions
import supervision as sv

# create a simple box annotator to use in our custom sink
annotator = sv.BoxAnnotator()

def my_custom_sink(predictions: dict, video_frame: VideoFrame):
    # get the text labels for each prediction, along with the confidence level
    labels = [f"{p['class']} ({p['confidence']*100:.2f}%)" for p in predictions["predictions"]]
    # print the labels to the console
    for label in labels:
        print(label)
    # load our predictions into the Supervision Detections api
    detections = sv.Detections.from_inference(predictions)
    # annotate the frame using our supervision annotator, the video_frame,
    # the predictions (as supervision Detections), and the prediction labels
    image = annotator.annotate(
        scene=video_frame.image.copy(), detections=detections, labels=labels
    )
    # display the annotated image
    cv2.imshow("Predictions", image)
    cv2.waitKey(1)

pipeline = InferencePipeline.init(
    model_id="identificacao-de-blocos/1",
    video_reference=2,
    on_prediction=my_custom_sink,
    confidence=0.90,  # Set the confidence threshold to 90%
)

pipeline.start()
pipeline.join()
```

Apêndice D

```python
from inference import InferencePipeline
from inference.core.interfaces.camera.entities import VideoFrame
import cv2
import numpy as np
import supervision as sv

# create a simple box annotator to use in our custom sink
box_annotator = sv.BoxAnnotator()

def my_custom_sink(predictions: dict, video_frame: VideoFrame):
    # get the text labels for each prediction
    labels = [p["class"] for p in predictions["predictions"]]
    # load our predictions into the Supervision Detections API
    detections = sv.Detections.from_inference(predictions)
    # annotate the frame using our supervision annotator
    image = box_annotator.annotate(
        scene=video_frame.image.copy(), detections=detections, labels=labels
    )

    # Draw keypoints and center of mass
    for prediction in predictions["predictions"]:
        if "keypoints" in prediction:
            keypoints = prediction["keypoints"]
            # Extract x, y coordinates of keypoints
            keypoint_coords = np.array([[kp['x'], kp['y']] for kp in keypoints])
            # Calculate center of mass
            center_of_mass = np.mean(keypoint_coords, axis=0, dtype=int)
            # Print object name and center of mass coordinates
            print(f"Object: {prediction['class']}, Center of Mass: {center_of_mass}")
            # Draw keypoints
            for keypoint in keypoints:
                keypoint_x = int(keypoint['x'])
                keypoint_y = int(keypoint['y'])
                color = (0, 255, 0)  # Green color for keypoints
                cv2.circle(image, (keypoint_x, keypoint_y), 5, color, -1)
            # Draw center of mass
            cv2.circle(image, tuple(center_of_mass), 5, (255, 0, 0), -1)

    # display the annotated image
    cv2.imshow("Predictions", image)
    cv2.waitKey(1)

pipeline = InferencePipeline.init(
    model_id="identificacao-dos-keypoints/1",
    video_reference=2,
    on_prediction=my_custom_sink,
    confidence=0.90,  # Set the confidence threshold to 90%
)

pipeline.start()
pipeline.join()
```

Apêndice E

```python
import cv2
import numpy as np
from inference import InferencePipeline
from inference.core.interfaces.camera.entities import VideoFrame
import supervision as sv
import threading

# Create a simple box annotator to use in our custom sink
box_annotator = sv.BoxAnnotator()

# Define a shared state to store predictions
shared_state = {
    "keypoints_predictions": None,
    "object_predictions": None,
    "lock": threading.Lock()
}

def my_custom_sink_keypoints(predictions: dict, video_frame: VideoFrame):
    # Create an empty image to draw on
    image = video_frame.image.copy()

    # Calculate center of mass
    for prediction in predictions["predictions"]:
        if "keypoints" in prediction:
            keypoints = prediction["keypoints"]
            # Extract x, y coordinates of keypoints
            keypoint_coords = np.array([[kp['x'], kp['y']] for kp in keypoints])
            # Calculate center of mass
            center_of_mass = np.mean(keypoint_coords, axis=0, dtype=int)
            # Draw center of mass
            cv2.circle(image, tuple(center_of_mass), 5, (255, 0, 0), -1)

    # Return the image with the center of mass drawn
    return image

def my_custom_sink_object(predictions: dict, video_frame: VideoFrame, keypoints_image: np.ndarray):
    # Get the text labels for each prediction including confidence
    labels = [f"{prediction['class']} ({prediction['confidence']*100:.1f}%)"
              for prediction in predictions["predictions"]]
    # Load our predictions into the Supervision Detections API
    detections = sv.Detections.from_inference(predictions)
    # Annotate the keypoints image using our supervision annotator,
    # the predictions (as supervision Detections), and the prediction labels
    image = box_annotator.annotate(
        scene=keypoints_image, detections=detections, labels=labels
    )
    # Display the annotated image
    cv2.imshow("Combined Annotations", image)
    cv2.waitKey(1)
```

Apêndice E

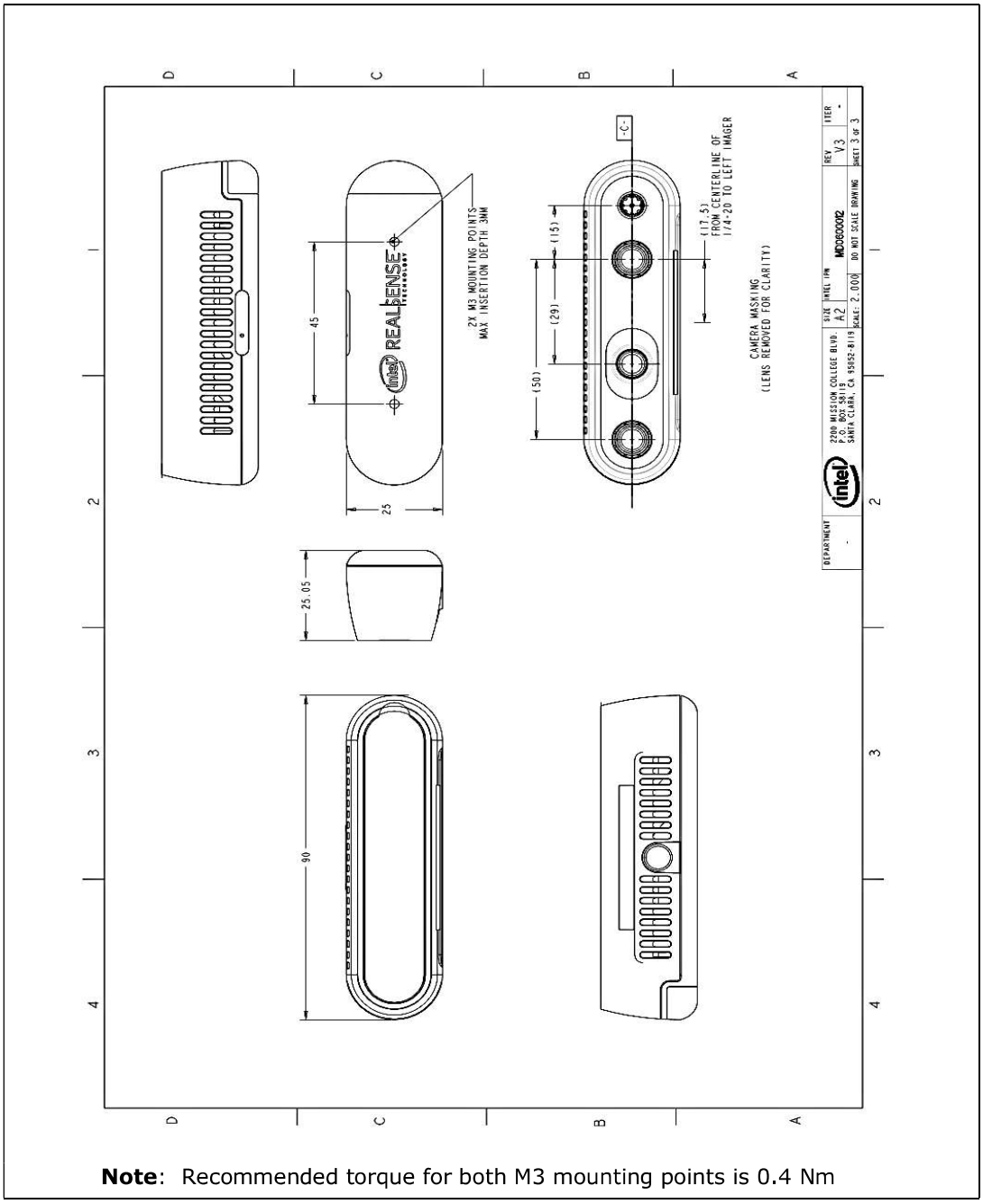
```python
def combined_sink(predictions_keypoints: dict, predictions_objects: dict, video_frame: VideoFrame):
    with shared_state["lock"]:
        if predictions_keypoints is not None:
            shared_state["keypoints_predictions"] = predictions_keypoints
        if predictions_objects is not None:
            shared_state["object_predictions"] = predictions_objects

        if shared_state["keypoints_predictions"] and shared_state["object_predictions"]:
            keypoints_image = my_custom_sink_keypoints(shared_state["keypoints_predictions"], video_frame)
            my_custom_sink_object(shared_state["object_predictions"], video_frame, keypoints_image)
            shared_state["keypoints_predictions"] = None
            shared_state["object_predictions"] = None

pipeline_keypoints = InferencePipeline.init(
    model_id="identificacao-dos-keypoints/1",
    video_reference=2,
    on_prediction=lambda predictions, frame: combined_sink(predictions, None, frame),
    confidence=0.90,  # Set the confidence threshold to 90%
)

pipeline_object = InferencePipeline.init(
    model_id="identificacao-de-blocos/1",
    video_reference=2,
    on_prediction=lambda predictions, frame: combined_sink(None, predictions, frame),
    confidence=0.95,  # Set the confidence threshold to 90%
)

# Start the pipelines in separate threads
keypoints_thread = threading.Thread(target=pipeline_keypoints.start)
object_thread = threading.Thread(target=pipeline_object.start)

keypoints_thread.start()
object_thread.start()

keypoints_thread.join()
object_thread.join()
```

Anexo A

Table 3-49. Depth Camera SKU Properties

D400 series Depth Cameras	Intel® RealSense™ Depth Camera D415	Intel® RealSense™ Depth Camera D435/D435i/D435f/D435if	Intel® RealSense™ Depth Camera D455 / D455f/ D456	Intel® RealSense™ Depth Camera D405
Depth module	Intel® RealSense™ Depth module D415	Intel® RealSense™ Depth module D430	Intel® RealSense™ Depth module D450	Intel® RealSense™ Depth module D401
Baseline	55mm	50mm	95mm	18mm
Left/Right Imagers Type	Standard	Wide	Wide	Wide
Depth FOV HD (16:9) (degrees)	H:65 / V:40 / D:72	H:87 / V:58 / D:95	H:87 / V:58 / D:95	H:84 / V:58 / D:92
Depth FOV VGA (4:3) (degrees)	H:50/ V:40 / D:61	H:75 / V:62 / D:89	H:75 / V:62 / D:89	N/A
IR Projector	Standard	Wide	Wide	N/A
IR Projector FOV	H:67 / V:41 / D:75	H:90 / V:63 / D:99	H:90 / V:63 / D:99	N/A
Color Sensor	OV2740	OV2740	OV9782	OV9782
Color Camera FOV	H:69 /V:42 /D:77	H:69 /V:42 /D:77	H:90 /V:65 /D:98	H:84 / V:58 / D:92
IMU	None	None/6DoF/None/6DoF	6DoF	None
Filter	All pass	All pass/ All pass/ IR-pass/IR-pass	All pass/ IR-pass	IR-cut

Figure 10-9. Intel® RealSense™ Depth Camera D435/D435i



Note: Recommended torque for both M3 mounting points is 0.4 Nm