Universidade Estadual de Londrina

ÍTALO STRESSER DOS SANTOS

INSPEÇÃO AUTOMATIZADA DE EQUIPAMENTOS DE PROTEÇÃO INDIVIDUAL BASEADA EM VISÃO COMPUTACIONAL

LONDRINA
2026

ÍTALO STRESSER DOS SANTOS

INSPEÇÃO AUTOMATIZADA DE EQUIPAMENTOS DE PROTEÇÃO INDIVIDUAL BASEADA EM VISÃO COMPUTACIONAL

Dissertação apresentada ao Programa de Mestrado em Engenharia Elétrica da Universidade Estadual de Londrina - UEL, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof(a). Dr(a). Maria Bernadete de Morais França

LONDRINA
2026

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Santos, Ítalo Stresser dos .
Inspeção Automatizada de Equipamentos de Proteção Individual Baseada em Visão Computacional / Ítalo Stresser dos Santos. - Londrina, 2026.
111 f.

Orientador: Maria Bernadete de Morais França.
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Estadual de Londrina, Centro de Tecnologia e Urbanismo, Programa de Pós-Graduação em Engenharia Elétrica, 2026.
Inclui bibliografia.

1. Visão Computacional - Tese. 2. Inspeção de EPIs - Tese. I. França, Maria Bernadete de Morais . II. Universidade Estadual de Londrina. Centro de Tecnologia e Urbanismo. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU 62

ÍTALO STRESSER DOS SANTOS

INSPEÇÃO AUTOMATIZADA DE EQUIPAMENTOS DE PROTEÇÃO INDIVIDUAL BASEADA EM VISÃO COMPUTACIONAL

Dissertação apresentada ao Programa de Mestrado em Engenharia Elétrica da Universidade Estadual de Londrina - UEL, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

BANCA EXAMINADORA

Orientador: Prof(a). Dr(a). Maria Bernadete de Morais França
Universidade Estadual de Londrina

Prof. Dr. Leonimer Flávio de Melo
Universidade Estadual de Londrina/
Departamento de Engenharia Elétrica

Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina/
Departamento de Computação

Londrina, 26 de fevereiro de 2026.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por me proporcionar as condições e oportunidades atuais.

Gostaria de expressar minha profunda gratidão à minha família pelo apoio incondicional, fundamental para que eu chegasse até esta etapa.

À minha orientadora, Maria Bernadete, registro meu agradecimento por sua constante disponibilidade em ajudar e pelos conhecimentos generosamente compartilhados ao longo desta jornada.

À Universidade Estadual de Londrina (UEL), instituição que me proporcionou a base da minha formação, agradeço pela estrutura, pelos ensinamentos e pelas oportunidades oferecidas.

Aos docentes, discentes e profissionais do Departamento de Engenharia Elétrica que, direta ou indiretamente, contribuíram para a realização deste trabalho e para meu crescimento pessoal e profissional.

Também pelo apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

SANTOS, I. S.. **Inspeção Automatizada de Equipamentos de Proteção Individual Baseada em Visão Computacional**. 2026. 111f. – Universidade Estadual de Londrina, Londrina, 2026.

RESUMO

Anualmente, milhões de pessoas sofrem acidentes ou doenças relacionadas ao trabalho, resultando em altos índices de mortalidade e afastamentos, tanto no cenário global quanto no brasileiro. Entre as medidas fundamentais para reduzir esses números está o uso adequado dos Equipamentos de Proteção Individual (EPIs). No entanto, a fiscalização presencial ou por videomonitoramento apresenta limitações que motivam a busca por soluções automatizadas. Diante disso, este trabalho desenvolve e avalia soluções voltadas à inspeção automatizada do uso de equipamentos de proteção individual, utilizando algoritmos de visão computacional aplicados à detecção do uso de capacetes de segurança em vídeos. O sistema emprega técnicas de filtragem temporal e rastreamento de objetos (BoT-SORT e StrongSORT) para mitigar falsos alarmes, enquanto um conjunto de dados foi compilado para o treinamento do modelo, visando aumentar sua precisão em cenários reais. Diferentemente de trabalhos que propõem novas arquiteturas de redes neurais ou novos conjunto de dados, este estudo concentra-se na implementação de mecanismos de filtragem de erros de detecção, na análise comparativa de algoritmos de rastreamento e na condução de experimentos sob restrições de hardware, buscando aproximar o sistema de cenários de aplicação real em monitoramento preventivo automatizado. Os resultados mostram que o modelo treinado com o conjunto de dados desenvolvido superou outros da área em métricas de mAP e também em testes externos, com cenários distintos das imagens de treinamento. A filtragem de falsos alarmes por quadros sequenciais resultou na mitigação do disparo de alarmes em trechos de detecções errôneas, tornando os alarmes mais confiáveis. De forma semelhante, o uso dos rastreadores contribuiu para a eficácia na mitigação de alarmes indevidos. Contudo, os resultados comparativos demonstram seu impacto no tempo de processamento e também sobre os algoritmos de reidentificação por CNN. Por fim, também foi possível observar desafios de detecção em situações como o uso de bonés e o distanciamento da câmera.

Palavras-chave: Detecção de EPI. Segurança no Trabalho. Dataset Capacetes. Automação de Inspeções

SANTOS, I. S.. **Automated Inspection of Personal Protective Equipment Based on Computer Vision**. 2026. 111p. – State University of Londrina, Londrina, 2026.

ABSTRACT

Annually, millions of people suffer from work-related accidents or illnesses, resulting in high rates of mortality and work absences, both globally and in Brazil. Among the fundamental measures to reduce these numbers is the proper use of Personal Protective Equipment (PPE). However, on-site inspection or video-based monitoring presents limitations that motivate the search for automated solutions. In this context, this work develops and evaluates solutions aimed at the automated inspection of personal protective equipment usage, using computer vision algorithms applied to the detection of safety helmet usage in videos. The system employs temporal filtering techniques and object tracking methods (BoT-SORT and StrongSORT) to mitigate false alarms, while a dataset was compiled for model training in order to increase accuracy in real-world scenarios. Unlike studies that propose new neural network architectures or new *datasets*, this work focuses on the implementation of detection error filtering mechanisms, the comparative analysis of tracking algorithms, and the execution of experiments under hardware constraints, aiming to approximate the system to real-world preventive monitoring applications. The results show that the model trained with the developed dataset outperformed others in the field in terms of mAP metrics and also in external tests, considering scenarios different from those present in the training images. False alarm filtering based on sequential frames mitigated alarm triggering in segments with erroneous detections, making the alarms more reliable. Similarly, the use of trackers contributed to the effectiveness of mitigating false alarms. However, comparative results also demonstrate their impact on processing time and on CNN-based re-identification algorithms. Finally, detection challenges were observed in situations such as the use of caps and increased distance from the camera.

Keywords: PPE Detection. Occupational Safety. Helmet Dataset. Inspection Automation

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

EPI	Equipamento de Proteção Individual
RFID	Radio Frequency Identification
IoT	Internet of Things
GPU	Graphics Processing Unit
CNN	Convolutional Neural Network
FPS	Frames Per Second
MOT	Multi-Object Tracking
ReID	Re-Identification
SGD	Stochastic Gradient Descent
IoU	Intersection over Union
mAP	mean Average Precision
AP	Average Precision
SORT	Simple Online and Realtime Tracking
PPG	Photoplethysmogram
LoRa	Long Range
YOLO	You Only Look Once
SSD	Single Shot Multibox Detector
SH	Safety Helmet
SFCHD	Safety Helmet and Head Detection Dataset
SHEL5K	Safety Helmet Dataset
COCO	Common Objects in Context
BoT-SORT	BoT-SORT multi-object tracking algorithm
StrongSORT	StrongSORT multi-object tracking algorithm
DeepSORT	DeepSORT multi-object tracking algorithm

SUMÁRIO

1 INTRODUÇÃO

Segundo estimativas da Organização Internacional do Trabalho, anualmente, 2,78 milhões de pessoas morrem devido a acidentes ou doenças relacionadas ao trabalho, e são registrados cerca de 374 milhões de acidentes não fatais (ILO, 2021). No Brasil, o Anuário Estatístico de Acidentes do Trabalho do Ministério da Previdência Social revelou um aumento no número de acidentes de trabalho entre os anos de 2021 a 2023, passando de 580.833 para 732.751 casos anuais (MPS, 2023).

Esses números evidenciam a necessidade de investigar e adotar estratégias eficazes para reduzir e mitigar esses casos. Entre as medidas fundamentais para garantir um ambiente de trabalho seguro, encontra-se o uso adequado dos Equipamentos de Proteção Individual (EPIs).

No entanto, como Baldissone *et al.* (2019) abordam, a principal barreira para a adoção dessa prática é a supervisão inadequada ou inexistente e, sem a devida fiscalização e correção, o comportamento inseguro tende a se consolidar como um hábito, aumentando os riscos para os trabalhadores.

Nesse contexto, a fiscalização do uso de EPIs pode ser realizada por meio de inspeções presenciais conduzidas por profissionais de segurança do trabalho, que verificam a conformidade diretamente nos locais de atividade laboral. Outra abordagem seria o videomonitoramento, permitindo a inspeção remota por meio de centrais de vídeo supervisionadas por operadores humanos. Entretanto, ambos os métodos apresentam limitações significativas: as inspeções presenciais demandam deslocamento contínuo de profissionais e podem não abranger todas as áreas de risco de forma eficaz. Já o videomonitoramento, embora amplie a capacidade de supervisão, ainda depende de observação humana contínua, tornando o processo repetitivo, sujeito a falhas e dispendioso em termos de horas de trabalho. Em ambientes industriais de grande porte ou canteiros de obras extensos, essas limitações tornam-se ainda mais evidentes.

Diante dessas limitações, observa-se o avanço de pesquisas voltadas à automação da inspeção de segurança do trabalho, que pode ser realizada por meio do uso de técnicas de visão computacional para a detecção do uso de EPIs. A automação da fiscalização surge como uma alternativa para reduzir a dependência da supervisão humana contínua, possibilitando o monitoramento em tempo real.

Contudo, é importante destacar que o foco do sistema proposto neste trabalho não está na fiscalização punitiva, mas sim na prevenção de acidentes. Busca-se aqui o desenvolvimento de uma solução capaz de atuar de forma proativa na segurança

ocupacional. Para isso, torna-se fundamental a automação do seu funcionamento, caracterizando-o como um sistema autônomo, capaz de operar continuamente sem intervenção humana direta e de emitir alertas preventivos em tempo real. Nesse contexto, a confiabilidade desses alertas é um requisito essencial.

Com esses objetivos, este trabalho propõe o desenvolvimento e a avaliação de uma abordagem voltada à detecção automatizada do uso de capacetes de segurança em vídeos, com ênfase na aplicação prática em cenários reais, avaliando algoritmos de visão computacional publicados na área. Para isso, são investigadas estratégias de treinamento de modelos, filtragem de detecções errôneas e uso de algoritmos de rastreamento, visando aumentar a confiabilidade dos alertas gerados e reduzir a ocorrência de falsos alarmes.

1.1 Objetivos

Este trabalho parte da hipótese de que a aplicação de filtragens temporais pode substituir o uso de rastreadores com ReID em cenários de monitoramento preventivo, mantendo desempenho adequado com menor custo computacional. Para isso, apresenta o estudo e o desenvolvimento de um sistema de visão computacional como tecnologia assistiva à segurança no trabalho. Para atingir esse propósito, foram definidos objetivos específicos que buscam suprir lacunas identificadas na literatura, presentes no Capítulo 2. Os objetivos são:

- Compilar, padronizar e testar uma versão complementar de conjunto de imagens existentes, de modo a formar um dataset generalista, balanceado e adequado ao treinamento de modelos destinados à detecção do uso e não uso de capacetes de segurança em cenários reais;
- Realizar análises em vídeos variando a taxa e quadros por segundo, avaliando a qualidade das detecções e suas limitações;
- Comparar diferentes estratégias na supressão de alarmes falsos, baseando-se em dois trabalhos apresentados no Capítulo 2.2.1, empregando algoritmos sem rastreamento, com rastreadores de múltiplos objetos com e sem mecanismos de reidentificação (*ReID*).

1.2 Organização do Trabalho

Este documento está organizado da seguinte forma: o Capítulo 2 apresenta uma revisão bibliográfica sobre tecnologias assistivas aplicadas à supervisão automática do uso de EPIs. O Capítulo 3 traz a fundamentação teórica de conceitos técnicos

utilizados ao longo do trabalho. No Capítulo 4, descreve-se a metodologia experimental, incluindo configurações, softwares e dados utilizados. O Capítulo 5 apresenta os resultados obtidos a partir dos experimentos realizados. Por fim, o Capítulo 6 expõe as conclusões.

2 REVISÃO BIBLIOGRÁFICA

Com o avanço de tecnologias digitais, surgem novas abordagens capazes de automatizar o processo de inspeção do uso de EPIS. Dentre essas tecnologias, destacam-se as soluções baseadas em sensores, como RFID (*Radio Frequency Identification*), infravermelho, IoT (*Internet of Things*) e as soluções com base em visão computacional.

2.1 Soluções Baseadas em Sensores

As soluções baseadas em sensores geralmente exigem a incorporação de dispositivos eletrônicos aos EPIs, de forma a permitir a identificação automática do seu uso por parte dos trabalhadores.

Um dos métodos envolve o uso de comunicação por radiofrequência (RFID), como apresentado por Kelm *et al.* (2013), que propuseram um portal móvel equipado com leitores RFID para verificar a presença dos EPIs. As etiquetas fixadas nos equipamentos permitiam a identificação em tempo real, conforme os trabalhadores atravessavam o portal.

Complementarmente, outras abordagens vêm explorando sensores infravermelhos e redes IoT. Por exemplo, Zhang *et al.* (2019) propuseram um sistema que detecta o uso de capacetes e envia alertas em tempo real para os gestores. Já **Battistoni** desenvolveram uma solução com dispositivos *Bluetooth* de baixo custo, operando em rede pessoal de proximidade.

Adicionalmente, Kim *et al.* (2021) implementaram um sistema baseado em comunicação LoRa, com sensores fotopletismograma (PPG) acoplados aos capacetes. Essa solução, voltada para ambientes com cobertura ampla, permite a verificação contínua do uso dos EPIs. Em uma abordagem semelhante, Abbasianjahromi e Ghazvini (2022) desenvolveram um dispositivo vestível conectado a uma rede IoT, promovendo o monitoramento em tempo real no canteiro de obras.

Apesar dos avanços, essas técnicas dependem do uso de dispositivos vestíveis pelos trabalhadores, o que exige adaptação e adesão ao seu uso, além da necessidade de incorporar sensores e módulos de comunicação nos equipamentos de proteção. Além disso, a estratégia de utilizar portais de detecção em pontos específicos, embora eficiente para monitoramento em áreas delimitadas, não garante a fiscalização contínua em outros momentos e locais de risco dentro do ambiente de trabalho.

2.2 Soluções com Visão Computacional

Aproveitando os avanços em redes neurais profundas e convolucionais demonstrados nas últimas décadas (LeCun; Bengio; Hinton, 2015), assim como a evolução dos recursos de *hardware* com GPU (*Graphics Processing Units*), surgiram abordagens que oferecem soluções mais eficazes e com menor dependência de acessórios para o monitoramento do uso de EPIs.

Em pesquisas e desenvolvimentos na área de visão computacional, os detectores de objetos têm se mostrado viáveis para múltiplas aplicações. As técnicas de detecção podem ser classificadas em estágio único ou duplo, sendo que os detectores de estágio único apresentam inferências mais rápidas, porém com menor acurácia (Zou *et al.*, 2023).

Na área de inspeção de EPIs, o detector de estágio duplo Faster R-CNN (Ren *et al.*, 2017) foi explorado devido à sua precisão, apesar de um tempo de inferência mais elevado. No entanto, para aplicações em tempo real, os detectores de estágio único, como as arquiteturas SSD (Liu *et al.*, 2016), YOLO (Redmon *et al.*, 2016) e suas versões mais recentes (Ali; Zhang, 2024), têm sido aplicados em estudos comparativos, aprimoramentos de algoritmos e no desenvolvimento de modelos para implementação prática.

Entre os diversos estudos e desenvolvimentos, observam-se investigações comuns entre diferentes publicações. Desta forma, para melhor organização e compreensão, esses padrões foram categorizados e apresentados na Tabela 1, que apresenta os principais eixos de pesquisa e desenvolvimento na detecção de EPIs por meio de técnicas de visão computacional. As categorias utilizadas para essa classificação são descritas a seguir:

Arquiteturas (1):

Alguns desenvolvimentos nesta área focam em melhorar as arquiteturas dos detectores, implementando algoritmos que apresentam melhores desempenhos nas métricas de validação, tanto na detecção dos objetos quanto na precisão das inferências.

Conjunto de dados (2):

Também se observa o trabalho de criação de conjunto de imagens para treinamento e validação, agrupando imagens e etiquetando os objetos alvos de detecção, como também utilizando técnicas de aumento de dados em imagens e melhorias que visam um melhor treinamento.

Método de inferência (3):

Pode ser observado que alguns autores buscaram analisar e testar diferentes formas de realizar a inferência, por exemplo, identificando as pessoas que estão com equipamentos de segurança, ou detectando-os separadamente e, em seguida, verificando se estão associadas ao corpo. Assim como existem análises que buscam desenvolver a detecção do risco de falta de EPI associado a situações durante o trabalho.

Tempo Real (4):

Alguns trabalhos fizeram implementações e testes com foco na capacidade de fazer detecções com recursos de *hardware* limitados, como dispositivos de computação de borda, ou também testes para operação em tempo real, como os casos de automação.

Prevenção (5):

Com algoritmos e *hardwares* que possibilitem a operação em tempo real, tornaram-se viáveis os trabalhos que exploram automações de prevenção e conscientização. Alguns estudos exploraram essa ideia e avançaram implementação e testes nessa área.

Tabela 1 – Classificação das referências segundo as cinco categorias de estudos e desenvolvimentos.

Referência	**1**	**2**	**3**	**4**	**5**
(Fang *et al.*, 2018)	✓	✓			
(Wu *et al.*, 2019)	✓	✓			
(Nath; Behzadan; Paal, 2020)		✓	✓	✓	
(Wang *et al.*, 2021)		✓			
(Otgonbold *et al.*, 2022)		✓			
(Ke; Chen; Guo, 2022)	✓	✓		✓	
(Vukicevic *et al.*, 2022)			✓		
(Xu; Huang; Huang, 2022)	✓		✓	✓	✓
(Hu; Ren, 2023)	✓				
(Lee; Yu, 2023)			✓		✓
(An *et al.*, 2023)	✓			✓	
(Lo; Lin; Hung, 2023)		✓			
(Lee; Choi; Choi, 2023)	✓		✓		
(Lema; Usamentiaga; García, 2023)			✓	✓	✓
(Ahmad; Rahimi, 2024)		✓			
(Di *et al.*, 2024)	✓	✓		✓	
(Lin, 2024)	✓			✓	
(Zaidi *et al.*, 2024)			✓		✓
ESTE TRABALHO				✓	✓

Como evidenciado na Tabela 1, os desenvolvimentos na área seguem diferentes vertentes. De forma adicional, estão listas as descrições resumidas das principais contribuições e investigações dos trabalhos citados e o motivo de distinção de contrução com este trabalho desenvolvido na Tabela 2.

Tabela 2 – Resumo de investigações/contribuições e distinções em relação a este trabalho.

Referência	**Investigação / Contribuições**	**Distinção com este trabalho**
(Fang *et al.*, 2018)	Método baseado em Faster R-CNN para detectar trabalhadores sem capacete em vídeos de vigilância em longa distância, incluindo comparações com RFID, Bluetooth.	Este trabalho utiliza uma arquitetura de estágio único (YOLO12s), mais adequada para operações em tempo real devido à maior velocidade de inferência, e foca em testes para de monitoramento preventivo.
(Wu *et al.*, 2019)	Realizaram implementações utilizando o detector SSD e um módulo para detecção automática do uso de capacete com identificação de suas cores. O estudo também disponibilizou publicamente o dataset GDUT-HWD, que possui imagens provenientes do SHD (ver Capítulo 4.1). O modelo SSD-RPA512 atingiu 83.89% de mAP.	Este trabalho difere ao empregar uma arquitetura mais recente (YOLO12 small) e focar na aplicação prática com testes para inserir automações preventivas de não uso de EPIs. Para isso abordou questões não presentes no trabalho de Wu *et al.* (2019), como integração de rastreadores, filtragem de erros e diminuição de FPS em vídeos.

Continua na próxima página

Tabela 2 – Resumo de investigações/contribuições e distinções em relação a este trabalho. (Continuação)

Referência	**Investigação / Contribuições**	**Distinção com este trabalho**
(Nath; Behzadan; Paal, 2020)	Proposição de três abordagens baseadas em YOLOv3 para detecção em tempo real do uso de capacete e colete, com foco na verificação de conformidade de EPIs. O estudo criou o dataset Pictor-v3 (1.500 imagens) e comparou métodos de detecção única, classificação direta e detecção + classificação por CNN. Com uma das abordagens obteve 72.3% de mAP e 11 FPS. O dataset e os modelos treinados foram disponibilizados publicamente.	Este trabalho não foca na comparação de múltiplas abordagens de inferência de uso de EPIs (ver Capítulo 4.1), mas na implementação prática com YOLO12 small, operações de filtragem, rastreamento e menor taxa de FPS, para automação preventiva. Além disso, utiliza um dataset voltado para detecção vestindo o EPI.
(Otgonbold *et al.*, 2022)	Proposição do dataset SHEL5K que é uma versão aprimorada e com anotações do dataset SHD. Utilizando como base a métrica mAP foram aferidas múltiplas arquiteturas YOLO (v3 a v5), Faster R-CNN e YOLOR, demonstrando suas melhorias. O dataset foi publicamente disponibilizado.	De forma complementar, este trabalho utiliza o *dataset* SHEL5K agregando a ele novas imagens e filtrando imagens não pertinentes. Além disso, ocorreram as implementações elencadas nos objetivos específicos deste trabalho com abordagens para viabilidade um sistema de automação com detectores de visão computacional.

Continua na próxima página

Tabela 2 – Resumo de investigações/contribuições e distinções em relação a este trabalho. (Continuação)

Referência	**Investigação / Contribuições**	**Distinção com este trabalho**
(Vukicevic *et al.*, 2022)	Avaliou seis arquiteturas de classificação de imagens (MobileNetV2, VGG19, DenseNet, SqueezeNet, InceptionV3, ResNet) em 18 tipos de EPI para cinco partes do corpo e utilizando estimativa de pose. Recomendou o MobileNetV2 por equilibrar desempenho e custo computacional. O conjunto de dados foi criado combinando imagens da web e datasets públicos.	Este trabalho difere por adotar uma abordagem de detecção unificada de EPI baseada em YOLO12 small, sem utilizar estimativa de pose. Enquanto Vukicevic *et al.* (2022) propõem um pipeline de duas etapas (pose + classificação por ROI), neste trabalho é empregado um detector de estágio único.
(Ke; Chen; Guo, 2022)	Proposição de um detector leve YOLOE para detecção de múltiplos EPIs com velocidade >100 FPS e utilizando GDUT-HWD *dataset* que é formado pelo SHD *dataset* (ver Capítulo 4.1)	Este trabalho foca na aplicação prática com baixo FPS, integração de rastreadores e filtragem de erros para automação preventiva, enquanto o estudo de Ke, Chen e Guo (2022) prioriza modificações em eficiência computacional para dispositivos de borda.
(Lee; Choi; Choi, 2023)	Proposição de uma abordagem em duas etapas (YOLOv5x + EfficientNetB7) para detecção de capacetes com distinção entre capacete, cabeça e chapéu. Foco na solução do problema de classificação errônea de chapéus como capacetes em ambientes reais.	Este trabalho adota uma abordagem de estágio único (YOLO12) para detecção unificada de EPIs, enquanto Lee et al. utiliza detecção seguida de classificação. Nosso estudo prioriza operação em tempo real com baixo FPS, integração de rastreadores e filtragem de erros para automação preventiva, além de que demonstra que com filtragem a incerteza de detecção de bonés fica menos relevante.

Continua na próxima página

Tabela 2 – Resumo de investigações/contribuições e distinções em relação a este trabalho. (Continuação)

Referência	**Investigação / Contribuições**	**Distinção com este trabalho**
(Lee; Yu, 2023)	Proposição de um processo ontológico de inferência que combina reconhecimento de objetos por IA com raciocínio baseado em relações para consciência de perigos em canteiros de obra. A abordagem define situações de segurança para inferir situações de trabalho atuais e identificar não conformidades com regras de segurança.	Este trabalho foca na detecção e verificação prática do uso de EPIs em tempo real, enquanto Lee e Yu (2023) investiga uma plataforma conceitual para inferência da situações de risco.
(An *et al.*, 2023)	Propoem uma versão melhorada do YOLOv5s, atingindo 92,40% de mAP@0.5 e 142,66 FPS, melhorando eficiência e velocidade para aplicações em tempo real.	Este trabalho utiliza uma arquiteturas mais recentes (YOLO12), explorando testes voltados à automação preventiva com testes em vídeos e comparações de filtragens para fins de sistemas autônomos.
(Lo; Lin; Hung, 2023)	Proposição de um modelo de detecção EPIs em tempo real baseado em YOLOv7, com comparações de desempenho com YOLOv3 e YOLOv4. O estudo construiu um novo *dataset* próprio (sem acesso público) para capacete e colete.	Este estudo enfatiza a integração de rastreadores e filtragem de erros para automação preventiva, enquanto Lo2023 compara arquiteturas YOLO em termos de precisão e velocidade sem abordar rastreamento ou filtragem de detecções sequenciais.

Continua na próxima página

Tabela 2 – Resumo de investigações/contribuições e distinções em relação a este trabalho. (Continuação)

Referência	**Investigação / Contribuições**	**Distinção com este trabalho**
(Hu; Ren, 2023)	Proposição do YOLO-LHD, uma versão aprimorada do YOLOv8n para detecção do uso de capacete, alcançando 94,3% mAP50 com apenas 0,86M de parâmetros.	Os objetivos deste trabalho diferem, pois não propõem novas arquiteturas, mas sim a análise prática de uso para automações preventivas, como em ocasiões de aplicação de detectores de EPIs e considerando os limites em custos computacionais.
(Ahmad; Rahimi, 2024)	Introdução do dataset SH17, com validação cruzada no conjunto externo Pictor-PPE, demonstrando a capacidade de generalização do modelo treinado.	O dataset criado possui abordagem distinta, focada na detecção unificada de EPIs, enquanto este trabalho utiliza um conjunto orientado à verificação da utilização de EPIs (ver Capítulo 4.1).
(Di *et al.*, 2024)	Proposição do detector MARA-YOLO baseado no YOLOv8-s, alcançando 74,7% de mAP no dataset KSE-PPE e operação em 80 FPS, além da elaboração de um novo dataset não público.	Este estudo concentra-se em testes práticos com YOLO12 small, comparação de rastreadores para filtragem de erros e preparação de conjuntos de imagens adequados para detecções em cenários reais.
(Lin, 2024)	Proposição do YOLOv8n-SLIM-CA, uma versão melhorada do YOLOv8n para detecção de uso de capacetes.	Este trabalho não foca na proposta de uma nova arquitetura de detecção, mas sim na análise prática de detecção de EPIs em tempo real com baixos FPS, integração de rastreadores e filtragem de erros para automação preventiva. Este estudo enfatiza a viabilidade operacional em restrições de hardware e cenários realistas com baixa taxa de quadros.

Continua na próxima página

Tabela 2 – Resumo de investigações/contribuições e distinções em relação a este trabalho. (Continuação)

Referência	Investigação / Contribuições	Distinção com este trabalho
Este trabalho	Implementação e testes com YOLO12 small focados na operação de um sistema de automação de alarmes. Integração de rastreadores e aplicação de filtragem de erros para permitir ações autônomas de prevenção. Também discute sobre redução de taxa de quadros por segundo para a implementação do sistema e filtragens.	Diferencia-se dos demais por concentrar-se na implementação de filtragem de erros, na análise comparativa de rastreamentos e na condução de experimentos realistas sob restrições de hardware. Em vez de propor novas arquiteturas ou novos datasets, investiga as lacunas necessárias para viabilizar a atuação autônoma em monitoramento preventivo.

2.2.1 Investigações a partir de trabalhos semelhantes

Após revisar trabalhos na área, como os apresentados na Tabela 2, alguns estão destacados nesta seção por abordar a aplicação dos detectores em cenários reais de automação. Eles são comentados a seguir, abordando suas contribuições e os contrastes de implementações e investigações com relação ao presente trabalho.

No trabalho de Xu, Huang e Huang (2022), os autores esboçaram um sistema preventivo de automação, utilizando múltiplas câmeras e minicomputadores interligados por rede sem fio para gerenciamento remoto, possibilitando tanto a geração de alarmes em tempo real no local quanto o envio de notificações aos gestores. Quanto às implementações, eles criaram o modelo Efficient-YOLOv5 e o testaram com imagens de canteiros de obras para aferir a qualidade das detecções.

O presente trabalho busca explorar a ideia proposta por Xu, Huang e Huang (2022). Para que essa proposta pudesse funcionar na prática, bastaria implementar o detector eficiente criado por eles em um software dedicado e executá-lo em um minicomputador de borda para gerar alarmes localmente?

A aferição do desempenho em vídeos de cenários externos aos conjuntos de treinamento pode ser uma investigação reveladora, sendo essa uma das fontes de estudo deste trabalho. Além disso, considerando aplicações reais, a implementação de filtragens se torna necessária para que um sistema seja autônomo e opere gerando alarmes de forma confiável.

Nessa linha de aplicação voltada à automação, Lema, Usamentiaga e García (2023) consideraram a necessidade de filtrar as detecções e possíveis erros. Os autores utilizaram um YOLOv5 treinado no conjunto de imagens CHV e rastreamento com StrongSORT, possibilitando, assim, uma filtragem individualizada. A solução foi baseada em minicomputadores (Jetson Nano) com câmeras dedicadas acopladas. Avançando para um nicho de estudo mais específico, essa abordagem de Lema, Usamentiaga e García (2023) foi investigada neste trabalho de forma a comparar a necessidade de utilizar rastreadores na solução em questão. Dessa forma, surgiu o seguinte questionamento: há a necessidade de realizar a filtragem com o StrongSORT, que utiliza uma CNN e onera consideravelmente o processamento dos quadros de imagem? Ou seria possível aplicar outro algoritmo menos impactante em termos de custo computacional, sem afetar o desempenho?

Pensando nessa questão, foram implementados os algoritmos StrongSORT e BotSORT, sendo este último aplicado com e sem a utilização da etapa de identificação por CNN.

Os autores, Zaidi *et al.* (2024), realizaram as filtragens de erros sem rastreador, utilizando um buffer de quadros de imagem e medindo a consistência de positivos de não uso de EPIs. Seria esse método de Zaidi *et al.* (2024) uma boa opção ao rastreamento de Lema, Usamentiaga e García (2023) ?

Esse método sem rastreadores foi aplicado ao nosso trabalho como uma alternativa a solução aplicada por Lema, Usamentiaga e García (2023).

Essas contribuições buscam possibilitar a implementação de ferramentas tecnológicas abordadas em diversas pesquisas acadêmicas em práticas de sistemas autônomos de monitoramento de segurança do trabalho, com enfoque em soluções economicamente viáveis e capazes de operar com confiabilidade.

As configurações dos códigos de desenvolvimento e os métodos empregados, são apresentados no Capítulo 4. As bases teóricas necessárias para compreensão das análises e implementações encontram-se no Capítulo 3, a seguir.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos e fundamentos para contextualizar e apoiar o entendimento dos desenvolvimentos deste projeto. São expostos os princípios das tecnologias empregadas, métodos e ferramentas computacionais utilizadas.

Considerando que o trabalho envolve a aplicação de técnicas recentes, como a detecção de objetos em imagens para fins de automação utilizando visão computacional, a seguir iniciam-se as descrições fundamentais relacionadas a esse tema.

3.1 Detecção de Objetos com Visão Computacional

A visão computacional é uma área que abrange métodos e técnicas pelas quais sistemas de visão artificial podem ser aplicados em situações práticas, unindo engenharia e ciência (Davies, 2005). Para a sua utilização no monitoramento de ambientes, é necessário um processo que vai desde a aquisição de imagens por câmeras até o processamento por meio de algoritmos modernos de aprendizado de máquina. A seguir, serão descritas algumas etapas e suas principais características.

3.1.1 Aquisição e formatação da imagem

As câmeras digitais utilizam um sistema óptico para direcionar o feixe de luz que incide sobre a cena até a superfície sensível do sensor. Esse sistema pode ser composto por uma única lente ou por um conjunto de lentes. A Figura 1 ilustra, de forma simplificada, o processo de formação e digitalização da imagem a partir da entrada de luz pela lente. Os sensores responsáveis pela conversão da luz em sinais elétricos são, em sua maioria, dos tipos CCD ou CMOS, ambos amplamente utilizados em câmeras digitais (Litwiller, 2005).

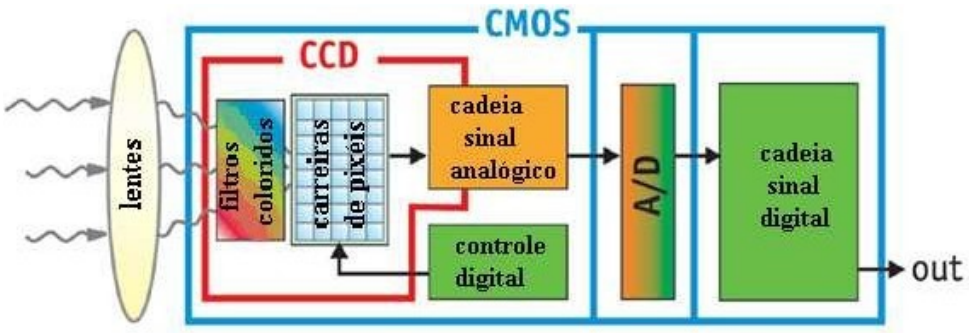


Figura 1 – Esquemático simplificado da arquitetura de funcionamento de câmeras digitais com sensores CMOS e CCD. Fonte: Adaptado de Litwiller (2005).

Os dados dos pixels quantizados pelo sensor podem ser representados conforme mostra a Figura 2, em três camadas RGB contendo valores de intensidade correspondentes a cada componente de cor. Essa representação matricial da imagem permite a aplicação de formatações de pré-processamento, tais como a conversão para monocromático, a normalização ou o redimensionamento, os quais contribuem para reduzir a complexidade computacional das etapas subsequentes de processamento (Elgendy, 2020).

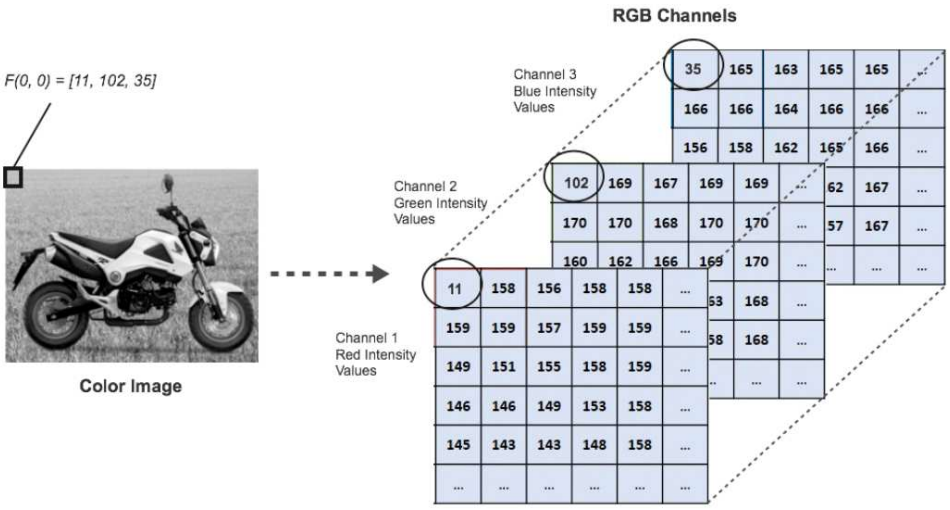


Figura 2 – Camadas de matrizes RGB com intensidades decimais para representação da imagem. Fonte: (Elgendy, 2020).

Com a imagem digitalizada em uma matriz de pixels, ela pode ser processada para extrair os recursos necessários às etapas subsequentes de análise computacional. O processamento de imagens busca aprimorar a qualidade e a representação dos dados visuais, seja para facilitar a interpretação humana ou para preparar a imagem para algoritmos de análise. Entre os conceitos fundamentais estão a vizinhança de pixels, os kernels de filtragem, a remoção de ruído e a detecção ou o realce de bordas (Solomon; Breckon, 2011).

Para a aplicação desses procedimentos pode ser utilizado a técnica de filtragem por meio de um *kernel* convolucional, que consiste em uma pequena matriz de valores que define o tipo de filtro a ser aplicado. O *kernel* é deslocado sobre a matriz de pixels, permitindo realçar características específicas, como bordas, texturas ou padrões. Diferentes *kernels* extraem diferentes tipos de características (Goodfellow; Bengio; Courville, 2016).

A Figura 3 ilustra a operação de convolução aplicada a uma imagem. Um *kernel* 2×2 percorre a matriz de entrada e, em cada posição, realiza multiplicações elemento a elemento seguidas de soma, produzindo um único valor na saída.

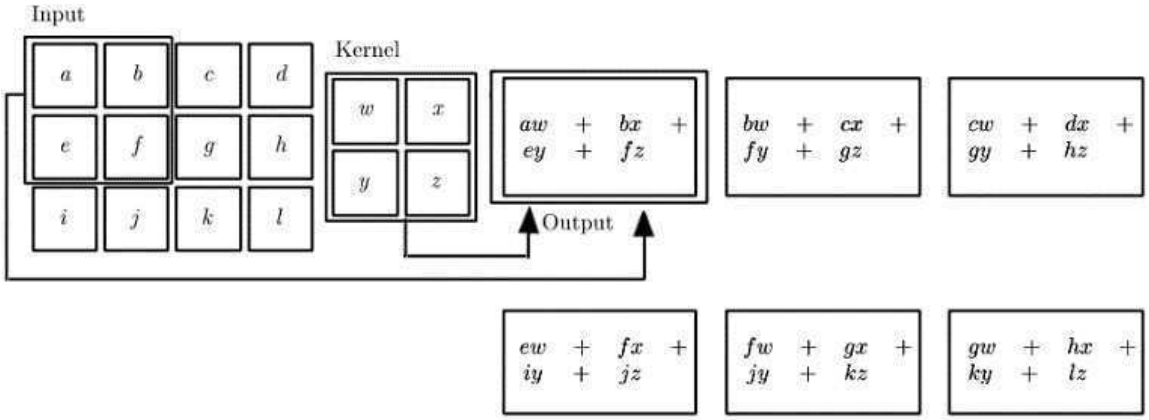


Figura 3 – Representação da aplicação do filtro de convolução em uma imagem. Fonte: Adaptado de (Goodfellow; Bengio; Courville, 2016)

3.1.2 Aplicação de Redes Neurais Convolucionais

De forma semelhante ao processo descrito na seção anterior, as Redes Neurais Convolucionais (CNN)(LeCun *et al.*, 1998) utilizam filtros convolucionais para aprender padrões visuais de maneira automática. Por meio de suas redes neurais de *deep learning*(LeCun; Bengio; Hinton, 2015), elas realizam o aprendizado de *kernels* para extração de características relevantes para identificação de objetos alvo.

As CNNs compartilham os pesos ao longo da imagem, o que torna seu processamento eficiente e reduz a sensibilidade à posição dos objetos. Com isso, um mesmo padrão visual pode ser reconhecido mesmo quando deslocado no quadro. Nas camadas iniciais, são extraídas características simples, que servem de base para que camadas mais profundas construam representações progressivamente mais complexas (Albawi; Mohammed; Al-Zawi, 2018).

Essas camadas são organizadas de modo que cada estágio da rede opere sobre os mapas de características gerados pelas convoluções anteriores. Em uma camada convolucional, cada filtro aprende um conjunto específico de pesos responsáveis por detectar um padrão local particular na imagem.

A organização dessas camadas, composta por conjuntos de mapas de características e filtros, foi apresentada por LeCun *et al.* (1998). Essa estrutura de camadas demonstra uma hierarquia na qual camadas subsequentes combinam e reprocessam os mapas produzidos nas etapas anteriores para gerar representações progressivamente mais complexas da imagem. Essa formação sequencial pode ser observada na arquitetura LeNet, introduzida por LeCun *et al.* (1998) e exibida na Figura 4.

Conforme ilustrado na arquitetura da Figura 4, as operações de convolução geram múltiplos mapas de características, que são subsequentemente reduzidos por operações de *subsampling*. Nas etapas finais, os mapas resultantes são aplanados e conectados a uma rede neural totalmente conectada (*Full Connection*), responsável

pela classificação final.

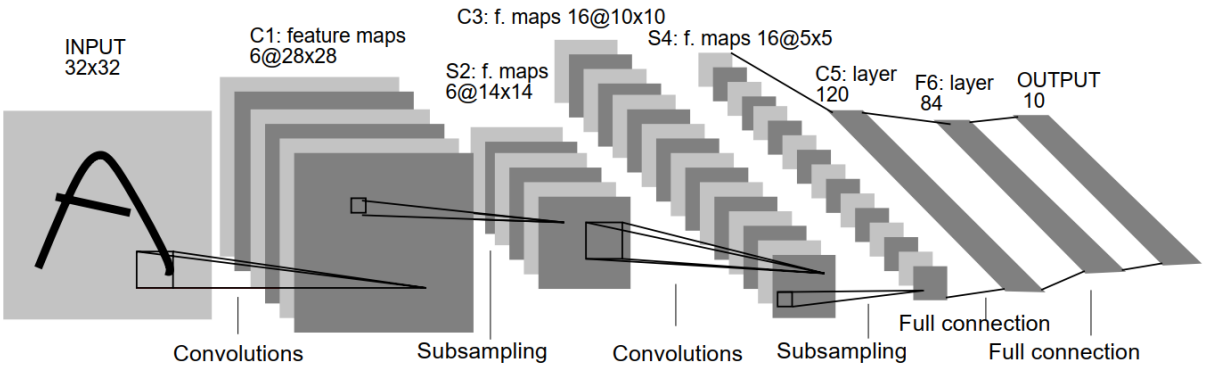


Figura 4 – Representação das camadas formadas pelas convoluções, as operações de *subsampling* e as camadas totalmente conectadas (*Full Connection*). Fonte: Adaptado de (LeCun *et al.*, 1998)

3.1.3 Detector YOLO

O YOLO (*You Only Look Once*) foi introduzido por Redmon *et al.* (2016) e possui em sua arquitura a CNN. Sua abordagem de detecção de objetos é realizada mapeando pixels da imagem diretamente para coordenadas de caixas delimitadoras e suas probabilidades de classe. Esse método resulta em tempos de inferência reduzidos quando comparado aos detectores de dois estágios, como a família R-CNN (Girshick *et al.*, 2014).

No caso dos detectores de dois estágios há um processo de detecção em duas fases distintas: primeiro geram propostas de regiões onde podem existir objetos e, em seguida, aplicam uma rede de classificação e regressão sobre cada proposta. O YOLO se diferencia por eliminar essa última etapa, realizando a detecção totalmente baseada em convoluções, o que reduz significativamente o custo computacional e permite inferências mais rápidas.

A arquitetura do YOLO pode ser descrita por três componentes principais:

- ***Backbone*:** Uma Rede Neural Convolucional responsável pela extração de características da imagem. Sua função é aprender representações discriminativas dos dados, conforme os princípios das CNNs descritos anteriormente.
- ***Neck*:** Estrutura intermediária responsável por combinar mapas de características em diferentes escalas por meio de operações de *downsampling* e *upsampling*. Exemplos de módulos empregados neste estágio incluem FPN (Lin *et al.*, 2017), PANet (Liu *et al.*, 2018) e BiFPN (Tan; Pang; Le, 2020).

- ***Head*:** Camada final responsável pelas predições, estimando as caixas delimitadoras (centro, largura e altura), a confiança de detecção e as probabilidades de classe.

Esses componentes têm recebido sucessivos avanços à medida que novos métodos de detecção são desenvolvidos. Diferentes pesquisadores e organizações passaram a propor novas versões do YOLO, incorporando melhorias nas componentes dessa arquitetura.

Os trabalhos de Ali e Zhang (2024) e Sapkota *et al.* (2025) tiveram como objetivo de seus trabalhos a revisão das versões YOLO, dessa forma os autores discutem a evolução do YOLO até sua 11ª versão (2024) e 12ª versão (2025), analisando as características da arquitetura, desempenho e modificações ao longo da evolução.

Conforme Sapkota *et al.* (2025), o YOLOv12, versão utilizada neste trabalho, introduz dois avanços arquiteturais que afetam diretamente o *backbone* e o *neck*. Com essas melhorias o YOLOv12n, por exemplo, alcança um aumento de aproximadamente 2,1 pontos de mAP(ver Capítulo 3.3) em comparação com os modelos equivalentes YOLOv10n e YOLOv11n, mantendo velocidade similar.

Conforme Sapkota *et al.* (2025), o YOLOv12, versão utilizada neste trabalho, introduz avanços arquiteturais que afetam diretamente o *backbone* e o *neck*. Com essas melhorias, o YOLOv12n, por exemplo, alcança um aumento de aproximadamente 2,1 pontos de mAP (ver Capítulo 3.3) em comparação com os modelos equivalentes YOLOv10n e YOLOv11n, mantendo velocidade similar.

Diante dessa evolução contínua das arquiteturas YOLO, a escolha do modelo adotado neste trabalho baseia-se no compromisso entre desempenho e custo computacional. Nesse sentido, optou-se pelo YOLOv12 na variante *small*, por incorporar melhorias recentes em relação a versões amplamente utilizadas na literatura, como o YOLOv5 e o YOLOv8. A escolha da variante *small* justifica-se pela necessidade de manter tempos de inferência reduzidos, aspecto fundamental para aplicações com restrições computacionais e potencial execução em tempo real, como o cenário considerado neste trabalho.

3.1.4 Treinamento do YOLO

Para que um modelo de detecção seja capaz de identificar corretamente o uso de EPIs em imagens, é necessário submetê-lo a um processo de treinamento. Esse treinamento ajusta os pesos da rede de detecção, permitindo que o modelo aprenda a extrair características visuais relevantes e, com base nelas, localizar e classificar os objetos de interesse.

Para realizar o treinamento do detector, é necessário configurar diversos hiperparâmetros que influenciam a convergência de aprendizado dos padrões de treinamento ou a capacidade de inferências em diversas imagens. Esses hiperparâmetros controlam como os pesos neurais da rede são ajustados em resposta aos erros observados durante o treinamento, visando a minimização da função de perda e a prevenção de problemas como *overfitting* ou *underfitting* (Goodfellow; Bengio; Courville, 2016; Elgendy, 2020).

No caso do YOLO utilizado neste trabalho, foi empregado o modelo desenvolvido pela Ultralytics®, previamente treinado no conjunto COCO (*Common Objects in Context*) (Lin *et al.*, 2014). O COCO é um dos conjuntos de dados mais amplamente utilizados em visão computacional, contendo milhares de imagens anotadas com diferentes classes de objetos em cenários do cotidiano. Após o treinamento inicial no conjunto COCO, realiza-se o refinamento de pesos da rede, utilizando para isso um processo de aprendizado de *fine-tuning*. Em arquiteturas baseadas em CNNs, estudos como Yosinski *et al.* (2014) demonstram que as camadas iniciais dos *backbones* aprendem características visuais mais genéricas, como bordas, texturas e formas simples, e por isso tendem a sofrer alterações mínimas durante o ajuste fino. Em contrapartida, as camadas mais profundas da rede se adaptam de forma mais significativa às novas classes e ao contexto específico do problema.

Esse tipo de treinamento também é utilizado no YOLO. Por padrão, todas as camadas permanecem treináveis durante o *fine-tuning*. É possível escolher, caso necessário, o congelamento do backbone, por exemplo. Para isso utiliza-se o argumento *freeze* para indicar quantos *layers* da arquitetura deseja-se manter com pesos fixos.

Entre os principais hiperparâmetros utilizados tanto nas redes YOLO, como também em outras arquiteturas, destacam-se:

- **Taxa de aprendizado** (*learning rate*): controla a velocidade com que o modelo ajusta seus pesos durante a otimização. Taxas muito altas podem causar divergência, enquanto taxas muito baixas podem resultar em uma convergência excessivamente lenta.
- **Número de épocas** (*epochs*): define quantas vezes o conjunto completo de dados de treinamento é utilizado para atualizar os pesos do modelo. Em geral, mais épocas permitem que o modelo aprenda melhor, mas podem levar ao *overfitting*.
- **Tamanho do lote** (*batch size*): número de amostras processadas antes de atualizar os pesos da rede. Tamanhos menores introduzem mais ruído nas atualizações (o que pode ajudar na generalização), enquanto tamanhos maiores proporcionam atualizações mais estáveis, porém mais exigentes computacionalmente.

- **Otimizador** (*optimizer*): algoritmo responsável por atualizar os pesos da rede durante o treinamento com base no gradiente da função de perda. Entre os métodos mais utilizados destaca-se o *Stochastic Gradient Descent* (SGD) (Robbins; Monro, 1951), o RMSProp (Tieleman; Hinton, 2012), o Adam (Kingma; Ba, 2015) e o AdamW (Loshchilov; Hutter, 2019). A escolha do otimizador influencia diretamente a velocidade e a estabilidade do processo de treinamento.
- **Momento** (*momentum*): O permite que o otimizador utilize informações das atualizações anteriores dos pesos, reduzindo oscilações e acelerando a convergência quando o gradiente aponta consistentemente na mesma direção.
- **Decaimento de peso** (*weight decay*): termo de regularização que penaliza valores elevados dos pesos do modelo, reduzindo sua complexidade e auxiliando na prevenção do *overfitting*, além de melhorar a capacidade de generalização.

Para adequada configuração desses hiperparâmetros é necessária a experimetação e, há ainda outros hiperparâmetros que podem ser ajustados, a depender da necessidade de melhorias por ajustes finos ou, da arquitetura de detector em utilização. Mais descrições de hiperparâmetros da rede YOLO utilizada neste trabalho podem ser encontradas na página do fabricante, a Ultralytics ®(ULTRALYTICS, 2025).

3.1.5 Contextualização de detectores em trabalhos semelhantes

Dentre os trabalhos citados no Capítulo 2, destaca-se o estudo de Hu e Ren (2023). Nesse estudo, utilizando o *dataset* COCO (Lin *et al.*, 2014), foram comparadas diferentes versões do YOLO com algoritmos consolidados de outras arquiteturas, como Faster R-CNN e SSD. Os resultados apresentados na Tabela 3 evidenciam a superioridade dos modelos YOLO em termos de precisão média e tempo de inferência.

Tabela 3 – Desempenho em precisão e velocidade de inferência dos modelos de detectores (dados de Hu e Ren (2023)).

Modelo	**COCO AP_{50-95} (%)**	**Tempo de Inferência (ms)**
SSD300	42,6	11,7
Faster R-CNN	47,8	21,9
EfficientDet	49,9	34,9
YOLOv5s	54,1	9,8
YOLO-LHD	54,0	10,4

Em outro estudo, Lin (2024) avaliou arquiteturas da família YOLO no contexto da detecção do uso de capacetes de segurança. Nesse trabalho, foi utilizado o conjunto de imagens SHWD (Gochoo, 2021), criado a partir do conjunto SHD (ANDREWMVD, 2020), o mesmo que serviu de base para o dataset SHEL5K citado no Capítulo 4.1. Trata-se de um conjunto semelhante ao empregado para o treinamento e validação

das inferências deste trabalho. Alguns resultados foram selecionados para compor uma referência adicional de desempenho em termos de mAP@0.5, conforme apresentado na Tabela 4. Nessa Tabela é possível observar o desempenho de versões do YOLO, tomando como base a métrica mAP, amplamente utilizada no âmbito de avaliação de detecção de objetos.

Tabela 4 – Valores de mAP@0.5 dos modelos avaliados (dados de Lin (2024)) para inspeção de EPIs.

Algoritmo	**mAP@0.5 (%)**
MARA-YOLO*	0.95040
YOLOv8n-SLIM-CA	0.94361
YOLOv8l	0.94118
YOLOv5s	0.92015
YOLOv5n	0.91674
SSD (VGG)	0.90186

* Referente à validação com classe capacete realizado por Di *et al.* (2024).

3.2 Elaboração de *Datasets* de Imagens

A construção de um *dataset* adequado é uma das principais etapas no desenvolvimento de sistemas baseados em visão computacional. Os dados de treinamento determinam diretamente a capacidade do modelo em reconhecer padrões e generalizar para novos contextos.

Por meio de um treinamento amplo, com milhares de imagens e instâncias de determinado objeto alvo de aprendizado, pode-se criar modelos de detectores com redes CNN (*Convolutional Neural Network*) (Krizhevsky; Sutskever; Hinton, 2012; LeCun *et al.*, 1998) que possuam boa eficácia em situações práticas.

Conforme exposto nos trabalhos de Shorten e Khoshgoftaar (2019) e Torralba e Efros (2011), os principais requisitos para a elaboração de um *dataset* eficiente incluem: variedade de cenários, equilíbrio entre classes, baixo viés de ângulos, categorias, ou iluminação, assim como a qualidade das anotações. Esses fatores ajudam a evitar problemas como o *overfitting* (quando o modelo se ajusta demais aos dados de treinamento) e o *underfitting*(quando o modelo não aprende padrões relevantes)(Goodfellow; Bengio; Courville, 2016).

Para melhorar a capacidade de generalização do modelo é comum utilizar técnicas de aumento de dados. Conforme discutido por Shorten e Khoshgoftaar (2019) e Claro *et al.* (2020), essas técnicas aplicam transformações artificiais sobre as imagens originais, gerando novas amostras para o treinamento sem a necessidade de coletar dados adicionais. Esse processo ajuda a reduzir o risco de *overfitting*, no qual o modelo memoriza os dados de treinamento e não consegue generalizar para novas

situações. As transformações podem ser agrupadas em diferentes categorias, que incluem desde manipulações básicas de imagens até métodos baseados em aprendizado profundo, conforme ilustrado na Figura 5.

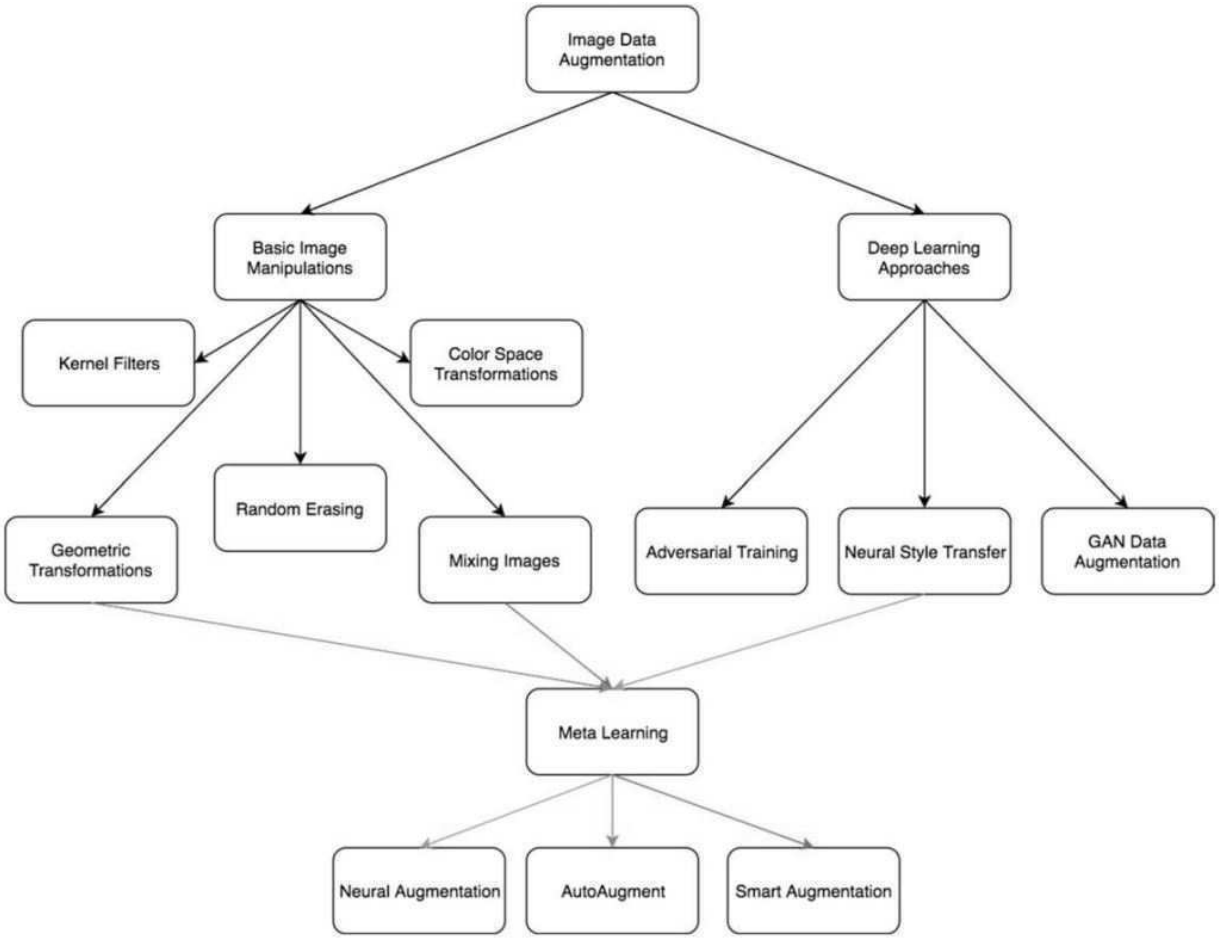


Figura 5 – Exemplos de métodos para aumento de dados em imagens. Fonte: Shorten e Khoshgoftaar (2019)

.

3.2.1 Anotação dos dados

A anotação dos dados consiste na delimitação precisa dos objetos de interesse por meio de caixas delimitadoras (*bounding boxes*) e na correta atribuição das respectivas classes. Esse processo pode ser realizado com ferramentas específicas, como o LabelImg (Tzutalin, 2015), ou por meio de plataformas online, como a Roboflow® (Dwyer; Nelson; Hansen, 2024), que, além da anotação, oferecem funcionalidades adicionais para o gerenciamento do conjunto de dados, balanceamento entre classes e aplicação de técnicas de aumento de dados (*data augmentation*).

A Figura 6 apresenta exemplos de caixas de detecção geradas pelo detector YOLO, evidenciando como os objetos são automaticamente localizados e classificados em quadros de imagem ou vídeo. A marcação correta dessas regiões é essencial para

validar a inferência do modelo, permitindo tanto a identificação de acertos e erros de detecção. que utilizam a correspondência precisa entre a área anotada e o objeto real presente na imagem.

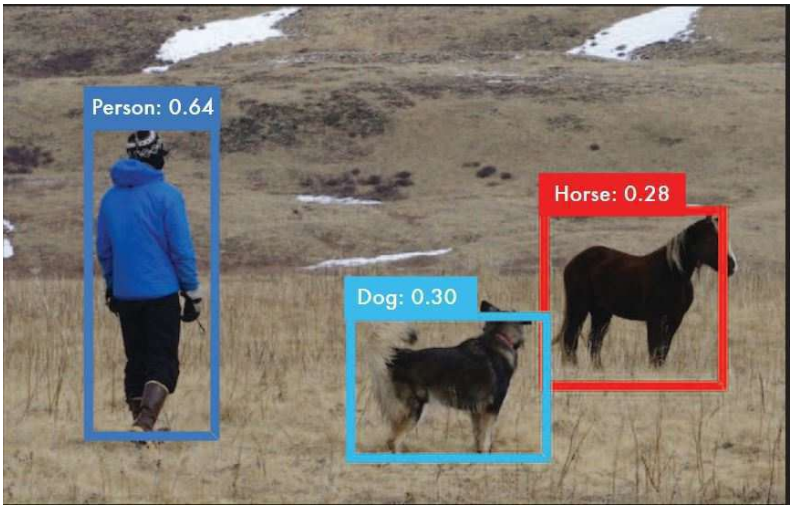


Figura 6 – Exemplos de caixas de detecção preditas pelo detector YOLO. Fonte: adaptado de Redmon *et al.* (2016).

3.3 Avaliação de sistemas de Detecção de objetos

Ainda sobre o treinamento do detector de objetos, é imprescindível o conhecimento das métricas que são utilizadas para comparar arquiteturas ou validar o desempenho do modelo criado. Essas métricas permitem avaliar não apenas a acurácia, mas também a robustez e a capacidade de generalização do modelo em diferentes cenários de teste.

Conforme descrito por Elgendy (2020) e Lin *et al.* (2014), algumas das métricas mais relevantes são:

- **Precisão** (*Precision*): representa a proporção de detecções corretas em relação ao total de detecções realizadas pelo modelo, conforme a Equação 3.1. Uma alta precisão indica que o modelo acerta na grande maioria das vezes em que detecta objetos de interesse, dessa forma cometendo poucos falsos positivos.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.1}$$

  onde: TP (*True Positives*): detecções corretas; FP (*False Positives*): detecções incorretas;

- **Revocação** (*Recall*): mede a proporção de objetos corretamente detectados em relação ao total de objetos existentes no conjunto de teste, conforme a Equação 3.2. Uma alta revocação indica que o modelo identificou grande parte das

instâncias presentes nas iamgens e assim comete poucos falsos negativos.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.2}$$

onde: FN (*False Negatives*): objetos não detectados.

- **IoU** (*Intersection over Union*): avalia a sobreposição entre a caixa predita P e a caixa real R, sendo definida como a razão entre a área da interseção e a área da união das duas caixas:

$$\text{IoU} = \frac{\text{Area}(P \cap R)}{\text{Area}(P \cup R)}. \tag{3.3}$$

  Esse índice expressa o quanto as duas caixas se sobrepõem, sendo um critério fundamental para determinar se uma detecção deve ser considerada correta.

- **mAP** (*mean Average Precision*): métrica utilizada para avaliar e comparar modelos de detecção de objetos. É obtida calculando-se a (*Average Precision (AP)* de cada classe ordenado-os por diferentes limiares de confiança e, a partir da curva resultante entre *Precision–Recall*, obter a área sob essa curva. O mAP é então a média das APs de todas as classes, podendo ser calculado para um único limiar de IoU (por exemplo, mAP@0.5).

- **F1-Score**: métrica que combina simultaneamente Precisão e Revocação, representando a média harmônica entre ambas. O F1-Score é definido conforme a Equação 3.4:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

  Uma pontuação alta indica que o modelo apresenta simultaneamente baixa taxa de falsos positivos e de falsos negativos.

- **Acurácia** (*Accuracy*): representa a proporção total de previsões corretas (incluindo acertos de detecção e acertos de não detecção) em relação ao número total de previsões realizadas. A acurácia é dada por:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

  onde TN (*True Negatives*) corresponde às regiões corretamente identificadas como não contendo objetos.

Uma forma de representar o desempenho de classificadores em tarefas de visão computacional é por meio da matriz de confusão. Essa matriz organiza as quantidades de acertos e erros cometidos pelo modelo ao comparar as predições com as classes verdadeiras.

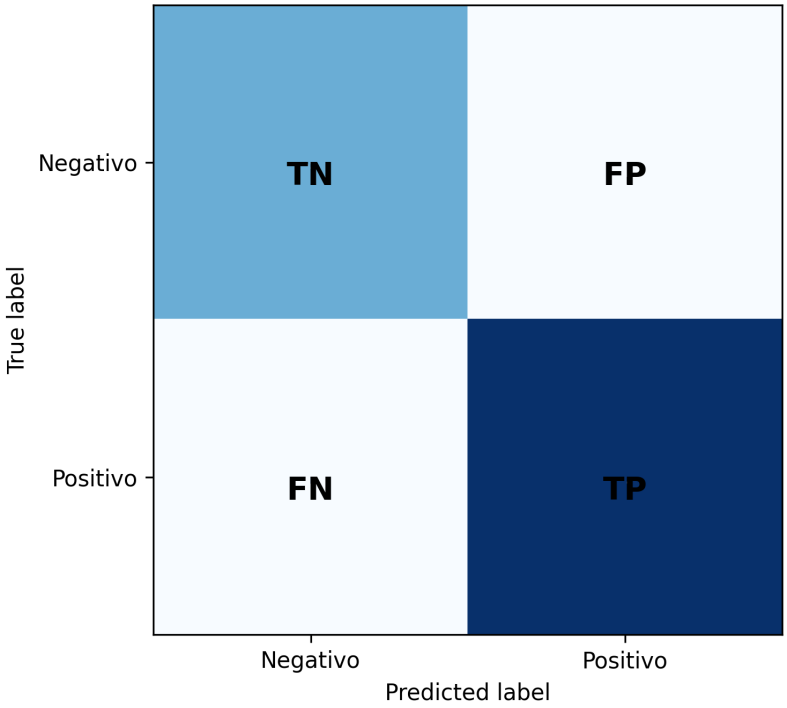


Figura 7 – Exemplos de matriz de confusão.

Na Figura 7 é apresentada a composição típica dessa matriz, evidenciando como são distribuídos os acertos verdadeiros e os diferentes tipos de erros (falsos positivos e falsos negativos) ao longo das categorias avaliadas.

Para aplicar essa representação à detecção de objetos, foi utilizada três classes, sendo uma classe de fundo que representa tudo aquilo que não são os objetos de interesse (com capacete ou sem capacete).

3.4 Rastreadores de Múltiplos Objetos

Um sistema de detecção de EPIs pode realizar inferências mais confiáveis ao verificar, em múltiplos instantes, se determinado indivíduo está utilizando corretamente seu EPI. Para isso, o uso de rastreadores de múltiplos objetos (MOT - *Multi-Object Tracking*) torna-se útil, pois permite acompanhar a trajetória de cada pessoa ao longo do tempo, associando as detecções realizadas em diferentes quadros do vídeo.

O desempenho dos algoritmos de rastreamento multi-objeto é frequentemente avaliado em MOT Challenge (Dendorfer *et al.*, 2021), que fornece comparações amplamente utilizadas na literatura. Nesse contexto, métodos como BoT-SORT e StrongSORT apresentam desempenho competitivo e esses foram utilizados em trabalhos desta área de detecção de EPIS. Eles estão descritos a seguir.

3.4.1 Algoritmos de rastreio

As abordagens de MOT utilizadas neste trabalho seguem os resultados de um detector de objetos que atua em cada quadro do vídeo, produzindo caixas delimitadoras (*bounding boxes*) para cada instância detectada. Em seguida, um módulo de rastreamento associa essas detecções ao longo dos quadros, formando trajetórias contínuas e preservando a identidade de cada objeto.

Entre os algoritmos representativos dessa abordagem destacam-se:

SORT(*Simple Online and Realtime Tracking*)(Bewley *et al.*, 2016): utiliza um Filtro de Kalman (Kalman, 1960) para prever a posição futura dos objetos e o Algoritmo Húngaro (Kuhn, 1955) para realizar a associação ótima entre as detecções e as previsões.

DeepSORT(Wojke; Bewley; Paulus, 2017): extensão do SORT que incorpora características visuais extraídas por uma rede de Re-Identificação (Re-ID), permitindo manter o rastreamento de objetos mesmo após oclusões parciais ou trocas de trajetória.

BoT-SORT (Aharon; Orfaig; Bobrovsky, 2022) : O BoT-SORT é uma evolução do DeepSORT que incorpora compensação de movimento de câmera, descritores de aparência mais robustos e ajustes nos parâmetros de associação, visando melhorar o desempenho em cenários densidade de objetos ou oclusões. Como opção, é possível selecionar o não uso do Re-Id(reindentificação via vetor de caraterísticas CNN), com isso escolhendo uma diminuição do custo computacional, porém com menor eficácia em cenários mais desafiadores para associação de instâncias.

StrongSORT (Du *et al.*, 2022): O StrongSORT também é uma evolução do DeepSORT, com foco em melhorias nos descritores de aparência e nos mecanismos de associação, buscando maior robustez e precisão em ambientes complexos de rastreamento.

3.4.2 Funcionamento da arquitetura

Os rastreadores destacados acima combinam os seguintes blocos em sua execução.

- **Estimação de estado:** prevê a posição futura de cada quadro e identificador a partir do estado atual. O filtro de Kalman é a técnica utilizada para essa predição/atualização.
- **Medida de associação:** define a correspondência entre detecções e rastros existentes. Essa etapa é resolvida pelo algoritmo Húngaro (Kuhn-Munkres)(Kuhn, 1955), que busca o custo mínimo de associação considerando diferentes métricas: distância de Mahalanobis(Mahalanobis, 1936), interseção sobre união (IoU) e semelhança de aparência obtida por vetores de características.

- **Modelo de aparência (ReID):** vetores de características extraídos por CNN permitem comparar a aparência entre uma detecção e as trajetórias anteriores. Essa técnica está presente nos algoritmos DeepSORT, StrongSORT e BotSORT, que possuem uma CNN pré-treinada para reidentificar as intâncias ao longo dos trajetos.
- **Gestão do rastreamento:** regras como, por exemplo, para iniciar, confirmar e encerrar rastros, ou escolher os limiares de nível de semelhança, formam hiperparâmetros específicos para cada algoritmo de rastreamento.

Esses algoritmos funcionam com predição de diferentes objetos ao decorrer de quadros de vídeos e suas versões de desenvolvimentos contribuíram para melhorias em sua tarefa de manter a identificação correta dos objetos. Ainda, nestes algoritmos, há hiperparâmetros ajustáveis para as necessidades do projeto. Para os dois rastreadores utilizados neste trabalho os parâmetros explorados por experimentações e seus valores configurados estão presentes no próximo capítulo de metodologia.

4 METODOLOGIA

Este capítulo apresenta os procedimentos metodológicos empregados no desenvolvimento do sistema de detecção e os testes efetuados.

As etapas de desenvolvimento foram divididas em cinco frentes, detalhdas nas seções 4.1 a 4.5 e resumidas conforme a seguir:

- Construção de um *dataset* complementar: foi compilado um conjunto de dados que amplia os disponíveis na literatura, incorporando características relevantes para o aprendizado de modelos de detecção, como diferentes ângulos de visão (como os de câmeras de vigilância), imagens de ambientes industriais e de construção civil ou presença de acessórios de cabeça. Além disso, visando reduzir viés de treinamento do modelo, buscou-se garantir um equilíbrio entre as classes de detecção,ampliando o número de instâncias da classe “sem capacete”;
- Treinamento do detector de EPIs: para possibilitar os testes de mitigação de alarmes falsos para automações preventivas em vídeos, foi necessário dispor de um modelo de detecção com desempenho semelhante ao estado da arte. Assim, desenvolveu-se um detector baseado na arquitetura YOLOv12, na versão *small*, capaz de atender aos requisitos de desempenho para os testes.
- Geração de vídeos e testes de detecção: foram criados e coletados vídeos contendo pessoas utilizando capacete ou não, considerando diferentes distâncias e cenários. Por meio desses vídeos foi avaliada a eficácia do detector YOLO12s treinados nos *datasets* distintos.
- Testes de erros de detecção e filtragens: Os vídeos foram aplicados no detector e no sistema com filtragem, sendo verificado os resultados com duas abordagens distintas, uma utilizando algoritmos de rastreamento de objetos e outra sem rastreamento.

As etapas de execução do sistema estão ilustradas na Figura 8 onde é possível identificar a representação dos bancos de imagens de diferentes fontes, assim podendo ser formado um *dataset* único para o treinamento do modelo de detecção.

4.1 Construção do *Dataset*

Nesta etapa, foi elaborado um conjunto de imagens para o treinamento do detector, a partir da coleta de dados de diversos *datasets* existentes. O objetivo foi compi-

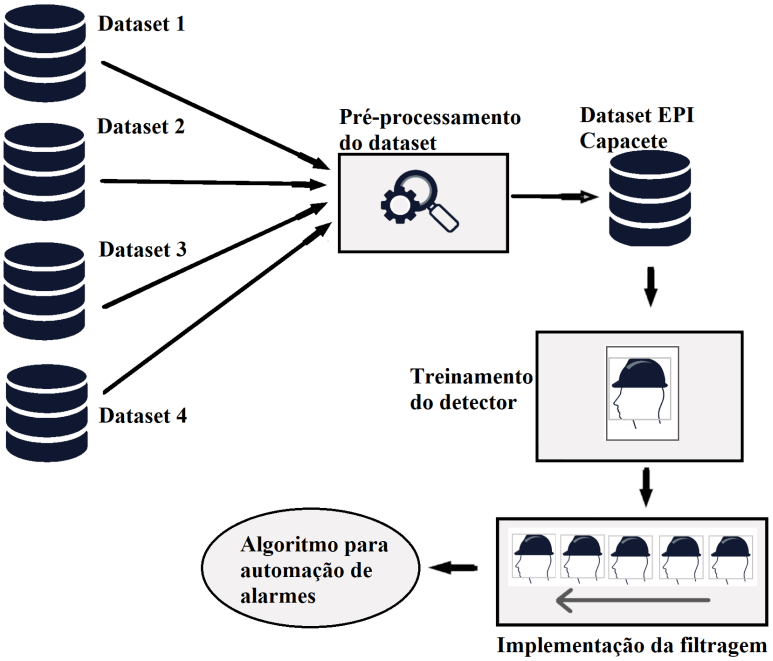


Figura 8 – Procedimentos para a elaboração do sistema de visão computacional para inspeção automática de EPIs. Fonte: do autor.

lar um número de instâncias superior ao disponível em conjuntos publicamente acessíveis, como os listados por Di *et al.* (2024): SWHD(Formado pelo SHD(Otgonbold *et al.*, 2022)), GDUT-HWD(Formado pela união do SHD e SCUT-Head(Peng *et al.*, 2018)), PictorV3, CHV, CHVG e FZU-PPE. Ainda mais, buscou-se construir um *dataset* mais generalista, contemplando diferentes cenários e variados ângulos de visão, reduzindo o risco de sobreajuste a contextos específicos, o que poderia comprometer o desempenho do detector em situações práticas.

Para o pré-processamento e formatação das imagens, utilizou-se a plataforma Roboflow® (Dwyer; Nelson; Hansen, 2024). As classes foram organizadas como “com capacete” e “sem capacete”. Essa abordagem, que identifica a ausência do EPI a partir da detecção da região completa da cabeça, foi empregada em trabalhos anteriores, como Nath, Behzadan e Paal (2020) e Otgonbold *et al.* (2022), e permite classificar se o indivíduo está utilizando o equipamento de proteção ou não.

4.1.1 Alterações para *augmentation*

Para aumentar a variabilidade visual dos conjuntos de imagens e aprimorar a capacidade de generalização do modelo, foram aplicadas técnicas de *data augmentation*, conforme Capítulo 3.2.

As transformações foram realizadas no ambiente de manipulação de imagens do Roboflow®. O objetivo não foi apenas expandir o número de amostras, mas também expor o modelo a uma variedade maior de condições visuais durante o treinamento.

Entre as possíveis transformações geométricas, foram selecionadas aquelas que aumentam a diversidade do conjunto sem comprometer a aparência realista das pessoas e dos capacetes. Assim, as modificações foram aplicadas de forma a evitar distorções excessivas que pudessem descaracterizar os objetos alvo de aprendizagem. As transformações utilizadas estão detalhadas a seguir:

- *Flip Horizontal*: consiste no espelhamento das imagens na direção horizontal. Inserido para que o modelo aprenda que a orientação lateral (esquerda ou direita) não é um fator determinante para a presença ou ausência do capacete.
- *Brightness*: ajuste do brilho das imagens entre -25% e +25%, a fim de simular diferentes condições de iluminação.
- *Blur*: aplicação de desfoque com raio de até 1 px, o que resulta em imagens com foco imperfeito, simulando condições de baixa nitidez.
- *Noise*: adição de ruído em até 1,5% dos pixels da imagem, o que contribui para aumentar a tolerância a imperfeições da captura de imagem.
- *Rotation*: rotação das imagens em torno do seu centro com variação de $\pm 15^{\circ}$, simulando ângulos de captura diferentes e aumentando a capacidade do modelo a mudanças de orientação do indivíduo com relação à câmera.

A partir dessas configurações, as transformações foram aplicadas a subconjuntos de imagens dos *datasets*. A quantidade final de imagens e instâncias por classe, após os pré-processamentos, pode ser verificada na Tabela 5. As fontes das imagens e o tratamento específico dado a cada conjunto são descritos a seguir.

Imagens do SFCHD

O *dataset* SFCHD, publicado por Yu *et al.* (2024), contém imagens anotadas de pessoas com e sem capacete, capturadas em ambientes industriais a partir de vídeos de câmeras de vigilância.

O conjunto original continha anotações de outras classes e imagens com marcações não pertinentes a este trabalho, as quais foram removidas.

Do conjunto resultante e utilizado para treinamento, foram selecionadas 200 imagens que continham a classe “cabeça” e a elas foram aplicadas as transformações de *augmentation*. Esse procedimento visou aumentar o número de instâncias classe “cabeça” e inserir novas imagens com caracteríticas distintas.

Imagens do CHV

Este conjunto *Color Helmet Vest* CHV (Wang *et al.*, 2021) contém imagens tanto de capacetes coloridos como de coletes de trabalho anotados. As imagens são de diversos contextos, com várias fotografias de construção civil e imagens com alta qualidade.

Desse *dataset*, foi extraído um subconjunto de 500 imagens. Como as anotações originais dos capacetes delimitavam apenas o equipamento e não a cabeça inteira, foi necessário retificar todas as instâncias dessa classe nas 500 imagens selecionadas.

A esse subconjunto reanotado, aplicaram-se as transformações de *augmentation*.

Imagens do SHEL5K

Este *dataset* (Otgonbold *et al.*, 2022) foi formado através de melhorias aplicadas ao *dataset* SHD(ANDREWMVD, 2020), incluindo alterações de *augmentations* geométricas, criação de novos *labels* e e retificação de *labels*.

Para a utilização dessas imagens, foram removidas as classes não pertinentes, assim como as imagens que não continham as duas classes de interesse. Nesse conjunto não foram aplicadas *augmentations* pois os autores já haviam implementado.

Imagens do CCTV-Head

Este é um projeto HeadDetectionCCTV (Then, 2023) está presente no Roboflow® (Dwyer; Nelson; Hansen, 2024). Desse conjunto, selecionou-se um subconjunto de 288 imagens, ao qual foram aplicadas as técnicas de *augmentation*.

Imagens do SCUT-Head

Este também é um projeto SCUT-HEAD (Peng *et al.*, 2018) que advém do Roboflow® (Dwyer; Nelson; Hansen, 2024) e possui um número total de 4.390 imagens e 109.788 instâncias de cabeças sem capacete. As imagens em sua grande maioria são de ângulo de câmeras de vigilância e possuem várias pessoas em cada quadro.

Desse conjunto, formou-se um subconjunto de 471 imagens. A partir desse subconjunto, foram aplicadas as técnicas de *augmentation* em 310 imagens.

Tabela 5 – Descrição quantitativa dos conjuntos de imagens formados após pré-processamentos.

Dataset	Quantidade de imagens	Instâncias com capacete	Instâncias sem capacete
SFCHD	11.692	14.291	1.478
CHV	896	1741	339
SHEL5K	4.956	16.051	6.120
CCTV-Head	1.012	0	4.853
SCUT-Head	781	0	17.169
SH	19337	29.328	30.559

4.1.2 Compilação após pré-processamento

Após a etapa de pré-processamento e aplicação das transformações de *augmentation*, cada um dos datasets estavam separados em subconjuntos de treinamento, validação e teste, com proporção 80:10:10. Com essa formatação eles foram unidos e compilados em um novo *dataset* nomeado SH.

Os conjuntos SHEL5K, SFCHD e CHV apresentavam predominância de instâncias da classe “com capacete”, o que poderia introduzir viés no treinamento do modelo. Além disso, no SFCHD observou-se uma escassa representação de indivíduos com diferentes adereços de cabeça e, no SHEL5K algumas imagens foram capturadas em contextos pouco realistas ou extraídas de fotografias disponíveis na web.

A distribuição final de instâncias por classe pode ser observada na Figura 9, que evidencia o melhor balanceamento e o maior número de instâncias após a compilação do novo conjunto.

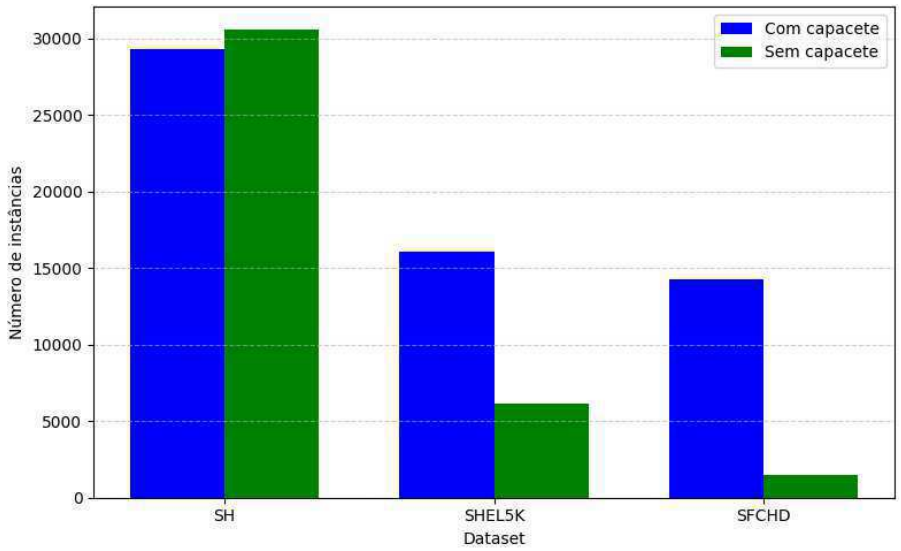


Figura 9 – Distribuição de instâncias por classe dos *datasets* SH, SHEL5K e SFCHD. As classes “com capacete” e “sem capacete” estão indicadas por cores distintas. Fonte: do autor.

4.2 Treinamento do Detector

Para o desenvolvimento dos modelos e a realização dos testes computacionais, foi utilizada a plataforma Google® Colab (*Colaboratory*). Esse ambiente disponibiliza recursos do Jupyter Notebook em instâncias com acesso a GPU NVIDIA® T4, o que permite reduzir o tempo de treinamento e de execução dos modelos.

Durante o treinamento e os testes do YOLO, foi utilizada a biblioteca Ultralytics ® (ULTRALYTICS, 2025), que fornece suporte à implementação e ao ajuste dos modelos de detecção. Essa biblioteca é desenvolvida a partir do *framework* de *Deep Learning* PyTorch, em Python. Para iniciar o treinamento, foi importado um modelo YOLO pré-treinado no *dataset* COCO (Lin *et al.*, 2014), empregando a técnica de *Transfer Learning* (ver Capítulo 3.1.4). Desse modo, os conjunto de imagens descrito no Capítulo 4.1 foram utilizados para o *fine-tuning* do detector.

A escolha da arquitetura do modelo de detecção influencia diretamente o desempenho do sistema. Assim, a análise de trabalhos relacionados, com verificação das arquiteturas empregadas, configurações de treinamento e resultados experimentais, é fundamental para o desenvolvimento de um modelo capaz de alcançar desempenho compatível com o estado da arte.

Desse modo, ao empregar um detector de última versão concebida com testes comparativos demonstrados em seu lançamento como o caso da versão YOLOv12 (Tian; Ye; Doermann, 2025), adotar configurações análogas às descritas na literatura, conduzir treinamentos com conjuntos de imagens semelhantes aos utilizados nas publicações da área, conforme exposto no Capítulo 4.1, ou até incluindo cenários mais desafiadores, como as imagens do conjunto SFCHD(Capítulo 4.1), e obter resultados de desempenho de treinamento melhores ou semelhantes aos trabalhos recentes na área (ver o Capítulo 3.1.5), a operação do algoritmo de detecção de capacetes empregado neste trabalho adquiri validade experimental.

Consequentemente, o detector de capacetes desenvolvido para atuação em testes de comparações de filtragens ou em aferições em cenários com novos vídeos, fornece evidências adicionais sobre o comportamento e a aplicabilidade no âmbito de sistemas automatizados para conformidade de uso de EPI's.

4.2.1 Evidências dos treinamentos com os *datasets* e o YOLO

Com base em estudos recentes na área de detecção de EPI's (Lee; Choi; Choi, 2023; Ahmad; Rahimi, 2024) e, considerando tanto o desempenho de arquiteturas distintas ou com versões anteriores, como descrito no Capítulo 3.1.5, foi selecionada a versão mais recente, YOLOv12 (Tian; Ye; Doermann, 2025).

Pela observação de trabalhos correlatos, iniciou-se a configuração do treina-

mento do modelo detector, observando o trabalho de Wang *et al.* (2021), que utilizou o YOLOv5 treinado em ambiente Ultralytics® com as bibliotecas PyTorch, empregando o otimizador SGD, taxa de aprendizado inicial de 0,01, *batch size* de 32 e pesos pré-treinados, além de dividir o conjunto de dados em 80% para treinamento, 10% para validação e 10% para teste. Também foi observado o trabalho de (Zaidi *et al.*, 2024), que comparou diferentes otimizadores (SGD, Adam, AdamW, AdaMax, NAdam e RAdam) utilizando YOLOv8 e assim mostrou os melhores desempenhos do SGD e do AdamW, com valores de *mAP* de 0.867 e 0.860, respectivamente, na validação.

Para a definição do modelo a ser aplicado nas etapas subsequentes, foram conduzidos treinamentos utilizando os otimizadores AdamW e SGD, com configurações de hyperparâmetros alteradas até que ocorresse a convergência do aprendizado ao longo das épocas. As medidas de *loss* e *mAP* obtidas durante o processo de treinamento do modelo com o conjunto SH, estão apresentadas nas Figuras 10 e11. Considerando o uso do modelo YOLO12 pré-treinado, com arquitetura e características para *trasnsfer learning* explicadas no Capítulo 3.1.4, foram realizadas 120 épocas de treinamento, número suficiente para estabilizar as curvas de perda, sem ganhos expressivos adicionais em *Precision* e *Recall*.

A escolha dos hiperparâmetros foi baseada nas configurações padrão recomendadas para modelos da família YOLO, sendo posteriormente ajustada empiricamente a partir do comportamento das curvas de treinamento e validação. Foram avaliados os otimizadores SGD e AdamW, considerando suas características de convergência e generalização, sendo selecionado, em cada caso, aquele que apresentou melhor compromisso entre redução de perdas e desempenho em mAP.

A seguir, na Tabela 6, são apresentados os hiperparâmetros utilizados para o modelo final utilizado nos testes em vídeos e mitigação de alarmes. Este modelo foi treinado com uso das configurações em modo padrão da Ultralytics, porém foi desabilitado no arquivo “build.py” a utilização automática de *augmentation* na épocas finais.

Tabela 6 – Hiperparâmetros utilizados no treinamento do modelo final.

Parâmetro	Valor
Modelo	YOLO v12 *small*
Epochs	120
Batch size	32
Lr0	0.01
Lrf	0.01
Image size	640x640
Dataset split	80:10:10
Optimizer	SGD
Momentum	0.937
Weight decay	0.0005

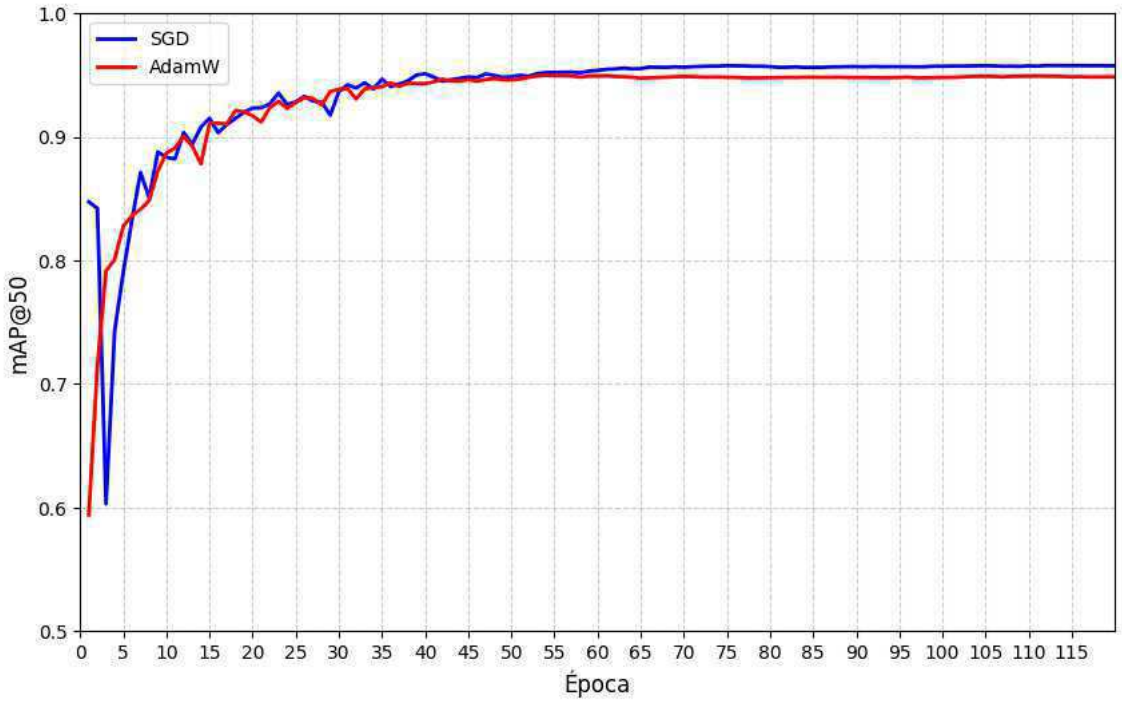


Figura 10 – Medidas de *mAP@0.5* durante o treinamento. Utilizando o YOLO12s, otimizadores SGD e AdamW e o *dataset* SH.

Além do conjunto SH, o modelo YOLO12s também foi treinado utilizando os datasets SFCHD e SHEL5K. As Figuras 12 13, 14 e 15 apresentam as curvas de perdas e a evolução da métrica mAP@0.5 no conjunto de validação. A otimização de pesos foi realizada com o SGD e AdamW e foram escolhidos os modelos com melhores resultados de diminuição de perdas e na métricas mAP. As tabelas 7 e 8 expõem a configuração de treinamento do modelos escolhidos para avançar em testes.

Tabela 7 – Hiperparâmetros utilizados no treinamento do modelo final com o SHEL5K.

Parâmetro	Valor
Modelo	YOLO v12 *small*
Epochs	77
Batch size	32
Lr0	0.0004
Lrf	0.01
Image size	640x640
Dataset split	80:10:10
Optimizer	AdamW
Momentum	0.9
Weight decay	0.0004

Os dados de treinamento, tais como as tabelas de perdas, métricas e modelos treinados, podem ser acessados pelo Anexo 6.1 deste documento.

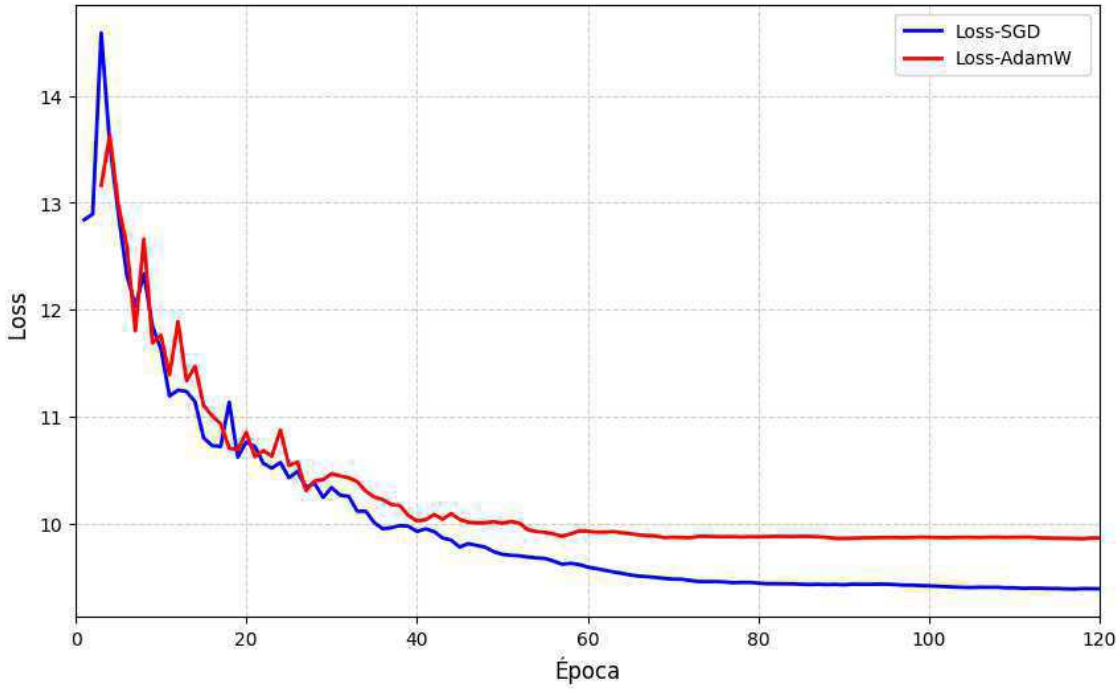
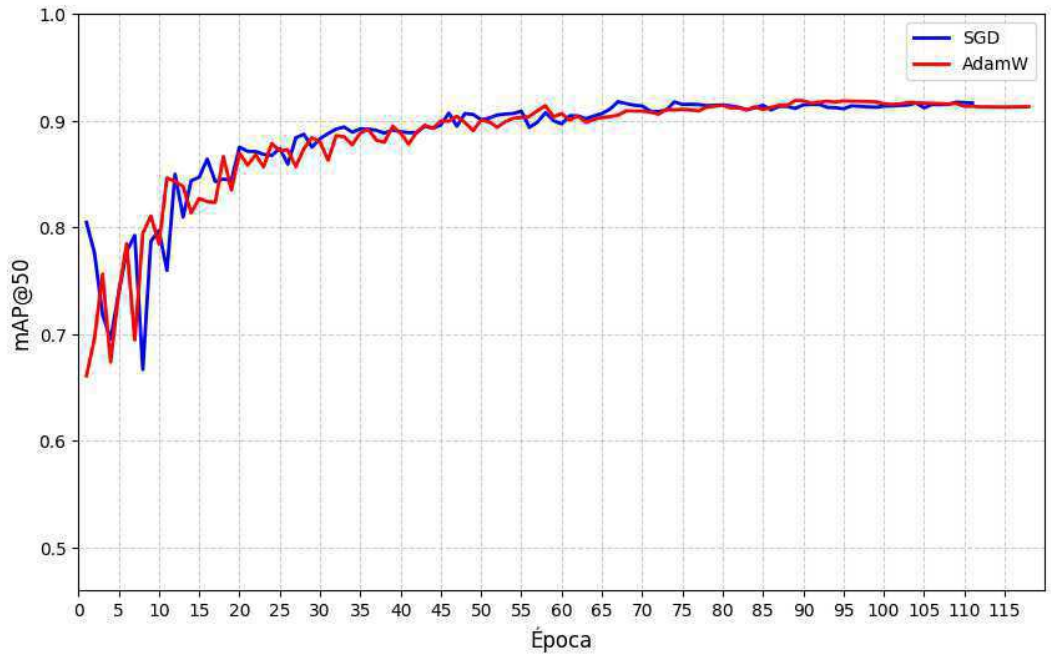


Figura 11 – Medidas de perdas durante o treinamento. Utilizando o YOLO12s, otimizadores SGD e AdamW e o *dataset* SH.

Tabela 8 – Hiperparâmetros utilizados no treinamento do modelo final com o SFCHD.

Parâmetro	Valor
Modelo	YOLO v12 *small*
Epochs	111
Batch size	32
Lr0	0.005
Lrf	0.01
Image size	640x640
Dataset split	80:10:10
Optimizer	SGD
Momentum	0.937
Weight decay	0.001



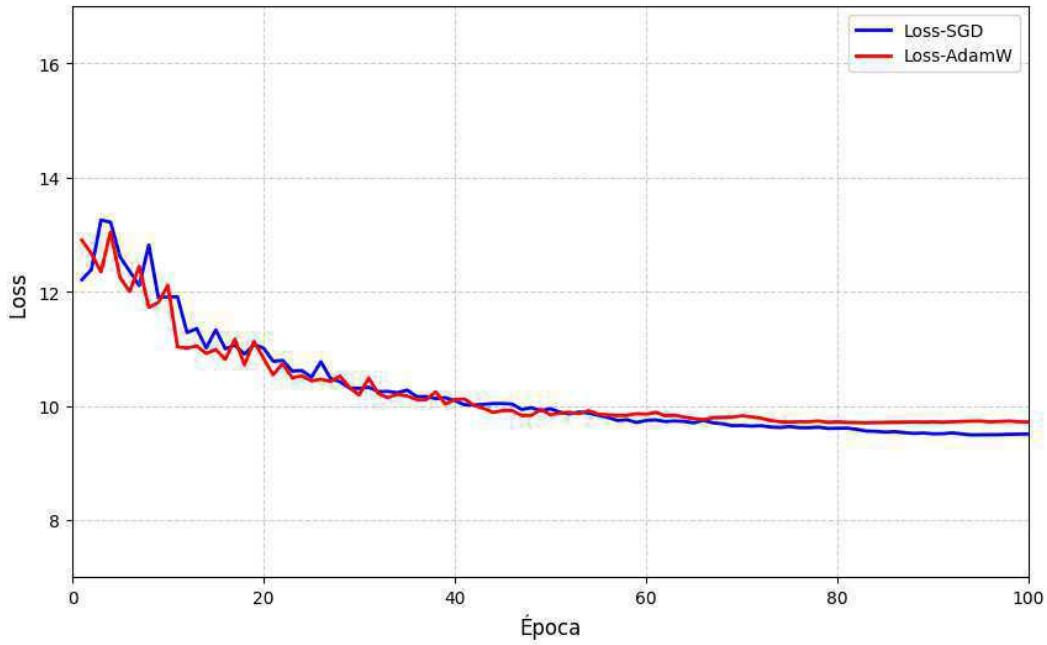


Figura 13 – Medida de perdas durante o treinamento com o *dataset* SFCHD, YOLO12s e otimizador SGD.

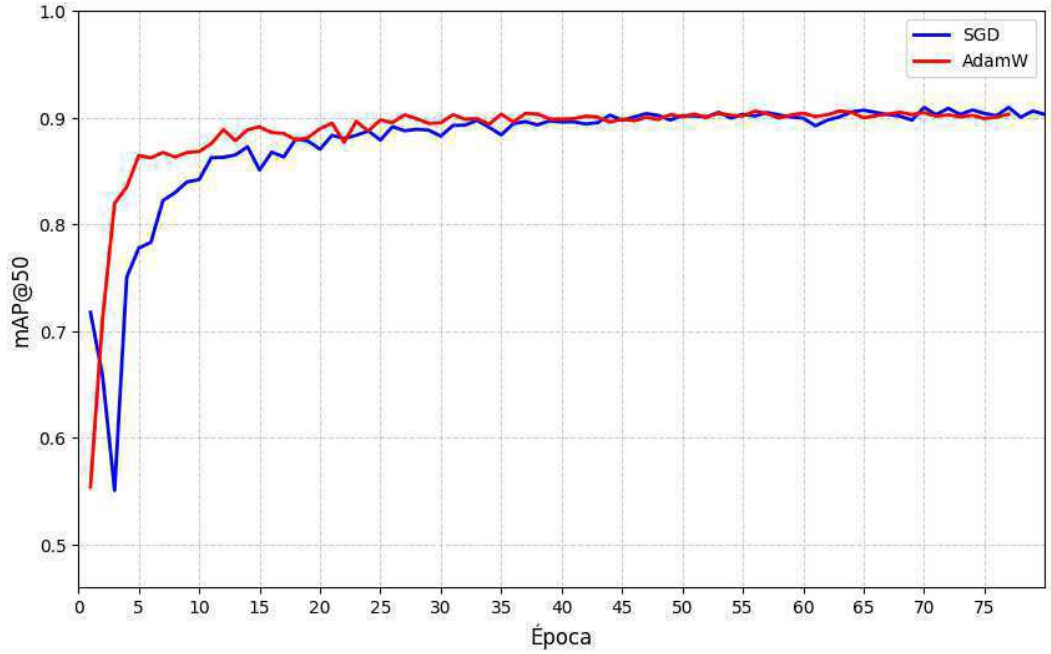


Figura 14 – Medida *mAP@0.5* durante o treinamento com o *dataset* SHEL5K, YOLO12s com o otimizador AdamW.

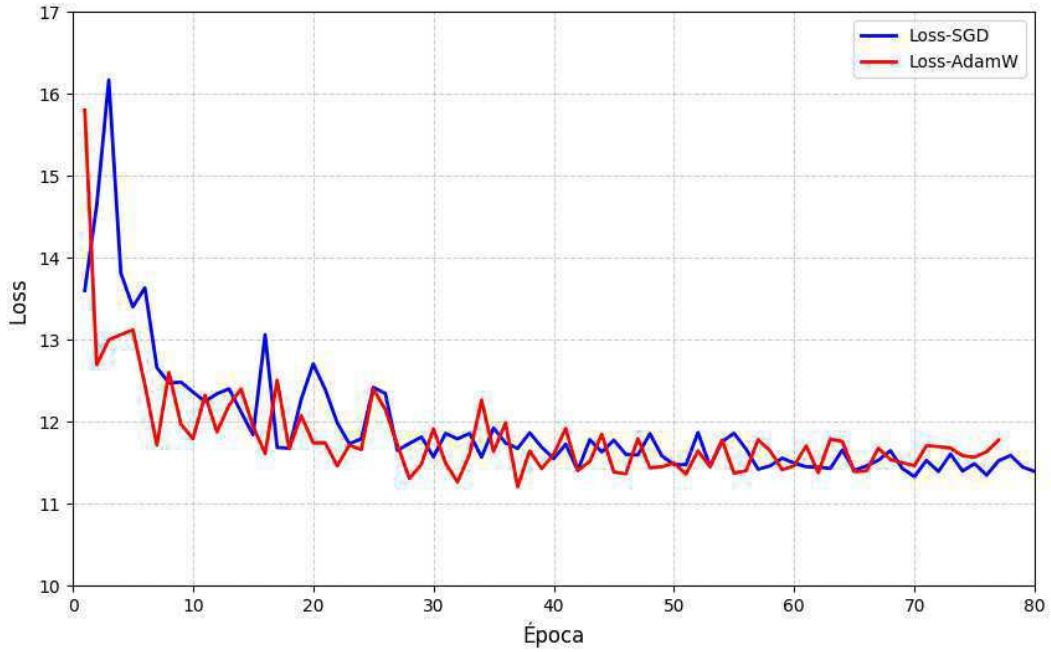


Figura 15 – Medida de perdas durante o treinamento com o *dataset* SHEL5K, YOLO12s e otimizador AdamW.

4.3 Algoritmos para supressão de falsas detecções

A fim de reduzir erros na inferência do uso de EPIs, foram implementadas duas abordagens de mitigação de falsos alarmes: uma baseada em *buffer* temporal de imagens e outra fundamentada em rastreadores de múltiplos objetos (MOT - *Multi-Object Tracking*).

4.3.1 Filtragem Temporal

A primeira abordagem de filtragem temporal, baseado em (Zaidi *et al.*, 2024), considera os resultados de detecção ao longo de uma janela deslizante, cuja dimensão é proporcional à taxa de quadros por segundo e ao tempo de inspeção definido para o local. Nessa abordagem, é mantida uma fila com os resultados de classificação de cada quadro, por exemplo, 30 resultados de quadros são considerados no cálculo de percentual.

A cada novo quadro processado, registra-se o valor `0` para ocorrências de não conformidade (sem EPI) e `1` para detecções conformes (com EPI). Ao final da janela, calcula-se o percentual de quadros com valor `0`. Caso esse valor ultrapasse um valor arbitrário de 70% um alarme é acionado.

4.3.2 Filtragem com Rastreadores MOT

A segunda abordagem de filtragem emprega algoritmos de rastreamento de múltiplos objetos (MOT), realizada de forma semelhante a Lema2023. Foram utilizados os rastreadores BoT-SORT e StrongSORT, disponibilizados por (Broström, 2023).

Esses rastreadores atribuem identificadores únicos (IDs) a cada objeto detectado e preservam suas respectivas classes, permitindo o acompanhamento individualizado ao longo do tempo. Essa abordagem reduz a interferência de detecções intermitentes ou instáveis, além de evitar falsos alarmes causados por objetos estáticos ou por mudanças momentâneas na visibilidade do capacete.

Para que o rastreador funcione em compatibilidade com o detector, os hiperparâmetros foram ajustados empiricamente e levaram em consideração a taxa de quadros por segundo e o grau de confiança das detecções da saída YOLO.

Os hiperparâmetros utilizados para o BoT-SORT com e sem ReID estão listados na Tabela 9 e descritos abaixo:

- `per_class`: Define se o rastreamento será realizado separadamente por classe. Neste trabalho não foi separado o rastreamento por classe para mante-lo mesmo após pequenas alterações de classe.
- `track_high_thresh`: Limiar de confiança para que um objeto seja considerado como parte do rastreamento.
- `track_low_thresh`: Um limiar intermediário para considerar como objeto candidadato ao rastreamento.
- `new_track_thresh`: Valor mínimo de confiança para iniciar um novo rastreamento (ID).
- `track_buffer`: Quantidade de quadros em que o objeto pode estar ausente antes do ID ser descartado.
- `match_thresh`: Limiar para associação entre detecções e rastros com base em medidas de similaridade.
- `proximity_thresh`: Distância máxima permitida entre detecções para associação inicial com base em localização.
- `appearance_thresh`: Limiar de similaridade visual extraída por reidentificação (ReID).
- `frame_rate`: Taxa de quadros usada para ajustar a dinâmica do rastreador ao vídeo processado.

- `fuse_first_associate`: Define se a fusão das informações de aparência e movimento é aplicada na primeira etapa de associação.
- `with_reid`: Indica se o módulo de reidentificação está habilitado. Para o BoT-SORT sem ReID é configurado como False.

Tabela 9 – Hiperparâmetros utilizados na configuração do rastreador BoT-SORT.

Parâmetro	**Valor**
`per_class`	`False`
`track_high_thresh`	0.5
`track_low_thresh`	0.5
`new_track_thresh`	0.6
`track_buffer`	5
`match_thresh`	0.6
`proximity_thresh`	0.4
`appearance_thresh`	0.7
`frame_rate`	30
`fuse_first_associate`	`False`
`with_reid`	`True`

De mesma forma, o rastreador StrongSORT foi configurado seguindo ajustes empíricos. A Tabela 10 apresenta os hiperparâmetros utilizados.

- `max_cos_dist`: Distância máxima entre vetores de aparência para associação por ReID.
- `max_iou_dist`: Limiar máximo de distância baseada em IoU (interseção sobre união) para associação.
- `max_age`: Quantidade de quadros em que o objeto pode estar ausente antes do ID ser descartado
- `n_init`: Número mínimo de quadros para confirmar um novo ID.
- `nn_budget`: Limite do buffer de vetores de aparência para cada ID. Valores altos podem impactar no custo computacional.
- `mc_lambda`: Peso de regularização do modelo de movimento.
- `ema_alpha`: Fator de suavização da média móvel exponencial dos vetores de aparência.

Tabela 10 – Principais hiperparâmetros utilizados no rastreador StrongSORT.

Parâmetro	**Valor**
`max_cos_dist`	0.3
`max_iou_dist`	0.7
`max_age`	8
`n_init`	3
`nn_budget`	10
`mc_lambda`	0.9
`ema_alpha`	0.4

4.4 Vídeos e imagens dos testes

Os vídeos utilizados foram obtidos por meio de extração da internet ou com autoria própria em locais do campus da universidade.

Para os vídeos realizados na universidade, foi utilizado um smartphone Samsung A15, que possui um sensor Hynix Hi5022Q e gravou os vídeos em resolução de 1280×720 pixels.

Para que o ângulo de captura se assemelhasse ao utilizado por câmeras de vigilância instaladas em locais de trabalho, foi utilizado um suporte com elevação em relação ao chão, ajustado entre 2,5 metros e 3,0 metros de altura.

Todos os vídeos tiveram o intuito de aferir possíveis erros de detecção e o comportamento da mitigação de alarmes falsos, para isso situações de diferentes cores de capacetes, tipos de iluminação, distâncias, ou número de pessoas foram alteradas, ampliando a diversidade de testes.

A seguir, são descritos os vídeos utilizados:

- **Vídeo 1:** Vídeo obtido de uma fonte da internet (VIDEEZY, 2025) e utilizado para testes tanto de detecções errôneas quanto de filtragem de erros. Ele foi armazenado em quadros de imagem de 640×640 pixels e duração de 6 segundos. O vídeo, gravado em ambiente externo e próximo a um porto, mostra um indivíduo que caminha em direção à câmera utilizando capacete e o retira por aproximadamente metade do tempo do vídeo.
- **Vídeo 2:** Gravado no campus da universidade, possuindo quadros de imagem de 1280×720 pixels e duração de 30 segundos, podendo ser acessado pelo Anexo 6.1. Para a gravação, a câmera foi posicionada a uma altura de 2,5 metros em um ambiente externo com iluminação natural. Neste vídeo, um indivíduo veste e também retira o capacete, caminhando em diferentes direções, variando a dis-

tância em relação a câmera entre 5 metros a 10 metros, inclinando o corpo e também se abaixando.

- **Vídeo 3:** Gravado no campus da universidade, possuindo quadros de imagem de 1280×720 pixels e duração de 10 segundos. Para a gravação, a câmera foi posicionada a uma altura de 2,90 metros em uma sala com diversos equipamentos e mesas e com iluminação artificial. No vídeo, dois indivíduos caminham ao redor de mesas de equipamentos, com uma distância em relação a câmera de 6 a 7 metros. Os capacetes utilizados foram de cores azul e amarelo e os indivíduos retiram e recolocam os capacetes.

- **Vídeo 4:** Gravado no campus da universidade, possuindo quadros de imagem de 1280×720 pixels e duração de 30 segundos, podendo ser acessado pelo Anexo 6.1. Para a gravação, a câmera foi posicionada a uma altura de 2,90 metros em uma sala com diversos equipamentos e mesas. Essa sala estava com luminosidade intencionalmente baixa. No vídeo, há um indivíduo com capacete branco que caminha em torno de uma mesa de equipamentos e retira e recoloca o capacete por alguns segundos. Ele caminha numa faixa de 4 a 5 metros da câmera.

- **Vídeo 5:** Gravado no campus da universidade, possuindo quadros de imagem de 1280×720 pixels e duração de 30 segundos, podendo ser acessado pelo Anexo 6.1. Para a gravação, a câmera foi posicionada a uma altura de 2,5 metros em um ambiente externo iluminação natural. Neste vídeo, um indivíduo utiliza boné durante todo o período do vídeo e caminha em diferentes direções, variando a distância em relação a câmera entre 5 metros a 10 metros, inclinando o corpo e também se abaixando.

- **Vídeo 6:** Gravado no campus da universidade, possuindo quadros de imagem de 1280×720 pixels e duração de 60 segundos. Para a gravação, a câmera foi posicionada a uma altura de 2,90 metros em uma sala com diversos equipamentos e mesas e, com iluminação artificial. No vídeo, existem dois indivíduos com capacetes de cor branca e azul e eles caminham em torno de uma mesa de equipamentos. Ambos os indivíduos retiram e recolocam os capacetes por alguns segundos. Eles caminharam numa faixa de 6 a 7 metros em relação à câmera.

- **Vídeo 7:** Gravado no campus da universidade, possuindo quadros de imagem de 1280×720 pixels e duração de 120 segundos. Para a gravação, a câmera foi posicionada a uma altura de 2,90 metros em uma sala com diversos equipamentos e mesas e, com iluminação artificial. De forma semelhante ao vídeo 6, existem dois indivíduos e desta vez os capacetes são de cores branca e amarela e eles

caminham por toda a sala, alterando a direção, passando um pela frente do outro e alterando a distância em relação à câmera numa faixa de 3 a 8 metros em relação à câmera. Ambos os indivíduos retiram e recolocam os capacetes por alguns segundos.

- **Vídeo 8:** Gravado no campus da universidade, possuindo quadros de imagem de 1280×720 pixels e duração de 30 segundos. Para a gravação, a câmera foi posicionada a uma altura de 2,5 metros em um ambiente externo com iluminação natural. Neste vídeo, um indivíduo retira o capacete por poucos segundos e depois caminha de uma distância de 3 metros a 10 metros percorrendo a região. Durante alguns segundos sai do quadro de vídeo e retorna e em outro momento se abaixa, ficando com o tronco na posição horizontal. A Figura 31 demonstra esses trechos citados e as detecções ou não detecções.

- **Vídeo 9:** Gravado no campus da universidade, possuindo quadros de imagem de 1280×720 pixels e duração de 60 segundos. Para a gravação, a câmera foi posicionada a uma altura de 2,90 metros em uma sala com iluminação artificial. Neste vídeo, um indivíduo retira e recoloca o capacete em alguns momentos e também caminha de uma distância de 6 metros a 12 metros.

- **Vídeo 10:** Gravado no campus da universidade, possuindo quadros de imagem de 1280×720 pixels. Para a gravação, foi utilizado um corredor com iluminação natural, apresentado na Figura 16, no qual a câmera foi posicionada a uma altura de 2,80 metros. Neste conjunto de vídeos, um indivíduo foi registrado caminhando em diferentes distâncias em relação à câmera, variando de 3 a 15 metros, com espaçamento de 3 metros entre cada posição, com o objetivo de avaliar o impacto da distância na qualidade das detecções.

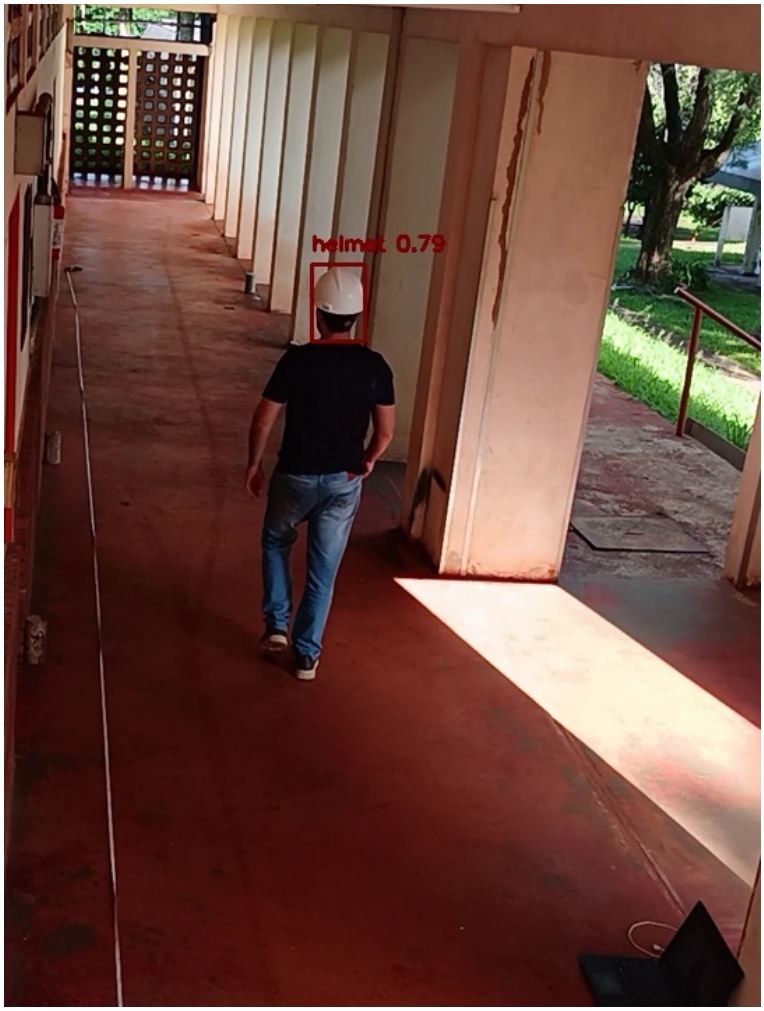


Figura 16 – Local utilizado para avaliar as confianças das inferências do detector YOLO12s-SH em função da variação da distância em relação à câmera.

Para que os resultados das inferências e da geração de alarmes possam ser analisados posteriormente, a Figura 17 ilustra por meio de linhas do tempo onde ocorreu o não uso do EPI e, também mostra os tempos os trechos com erros simulados e que serão descritos na Seção 4.4.2.

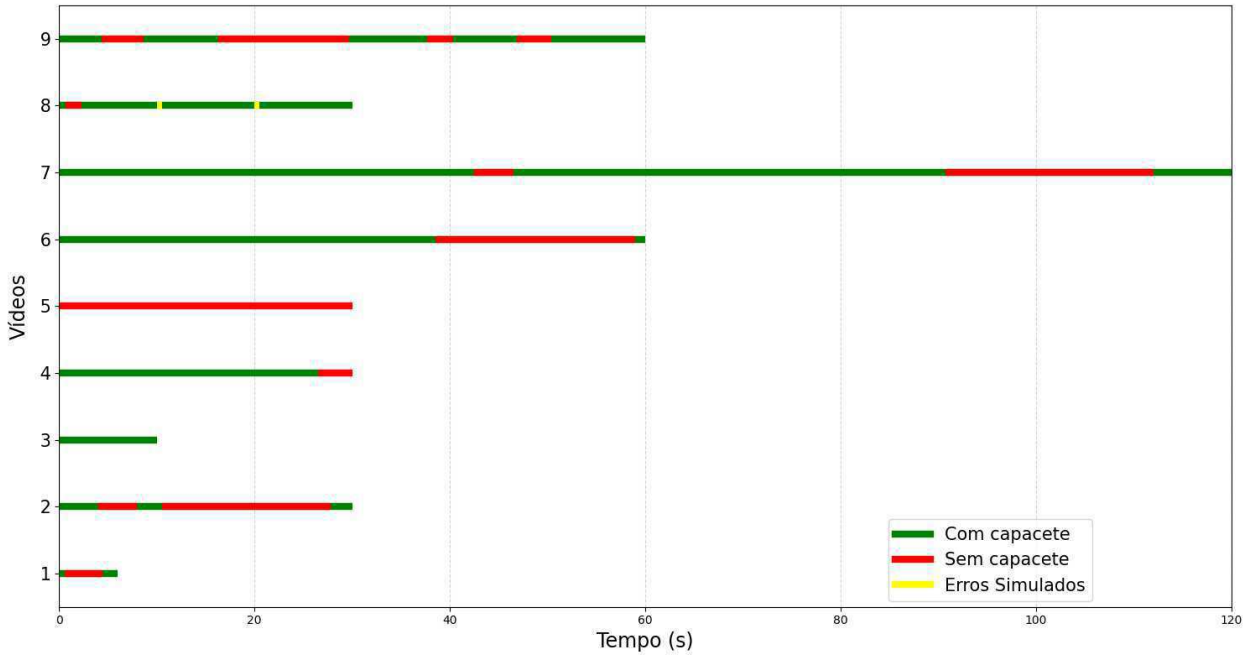


Figura 17 – Indicação dos valores reais do uso e não uso do capacete, com indicação de erros inseridos em trechos (amarelo) no decorrer do tempo dos vídeos

4.4.1 Taxa de Quadros por Segundo

Com o intuito de diminuir a carga de processamento das imagens, a taxa de quadros por segundo (*frames per second* - FPS) foi diminuída. Os vídeos originais foram convertidos para uma taxa quatro vezes menor, 7,5 fps. Essa conversão foi realizada utilizando a biblioteca `ffmpeg`(**ffmpeg**).

Essa etapa permitiu reduzir a quantidade de quadros processados por segundo, mantendo o conteúdo visual relevante para a detecção de eventos de não conformidade. Além disso, a redução de FPS viabiliza a execução de múltiplos fluxos de vídeo em paralelo com menor custo computacional, facilitando a futura aplicação do sistema em ambientes com múltiplas câmeras ou recursos de hardware limitados.

Os vídeos foram processados em duas taxas de quadros por segundo e a comparação manteve-se em relação a quantidade de alarmes e sua coerência com os real caso de não conformidade.

4.4.2 Erros de detecção

Os vídeos coletados foram submetidos ao detector e também aos módulos de rastreamento e filtragem. Porém, para a validação específica da capacidade de filtragem frente a erros na saída do detector, isto é, casos em que o YOLO produz falsos positivos de não uso de capacete (classe `0`), foram criados trechos artificiais no fluxo do algoritmo. Nesses trechos, indicados na Figura 17, o algoritmo de filtragem recebe propositalmente detecções alteradas, simulando respostas incorretas do detector ao

modificar a classe de detecção de `1` (com capacete) para `0` (sem capacete), além de ajustar a confiança para valores intermediários (entre 0,5 e 0,7). Essas variáveis alteradas foram a *class_id* e a *conf*, sendo realizadas diretamente na saída do YOLO, antes do processamento pelo módulo subsequente.

Os vídeos e simulações foram submetidos aos testes com os dois rastreadores e também ao modelo com filtragem sem rastreamento. Os resultados foram comparados e estão presentes no próximo próximo capítulo de resultados.

4.5 Testes de erros de detecção e filtragem

Com o objetivo de comparar o impacto da aplicação de diferentes estratégias de filtragem sobre o tempo de processamento e a geração de alarmes, foram desenvolvidos *pipelines* específicos para os quatro métodos avaliados: sem rastreador; com o rastreador BotSORT com ReID; com o rastreador BotSORT sem ReID e, com o rastreador StrongSORT. Cada um desses métodos foram realizados por cinco vezes para obtenção de um valor médio de resultado.

Os experimentos foram realizados na plataforma Google Colab, com acesso a uma GPU NVIDIA Tesla T4. O modelo YOLO12s-SH foi utilizado e o vídeo e modelo foram carregados para o ambiente de execução da instância do Colab. O vídeo foi executado dentro do mesmo *script* de testes, sendo que antes de cada vídeos foram aplicadas rotinas de liberação de cache da GPU e coleta de lixo da memória (torch.cuda.empty_cache() e gc.collect()), a fim de reduzir a interferência de resíduos de memória em cada execução.

O código completo utilizado para os testes encontra-se no Apêndice 6.1.

5 RESULTADOS

Nesta seção, são apresentados e analisados os resultados obtidos com a aplicação do modelo YOLO12s em vídeos, sob diferentes configurações de teste e métodos de mitigação de alarmes, com foco na avaliação do desempenho do sistema proposto.

5.1 Desempenho dos modelos treinados

Através dos detectores treinados foi possível avaliar seus desempenhos e verificar quais modelos obtiveram melhores performances em situações de vídeos de cenários distintos.

Inicialmente, para comparação, o cruzamento de aferições nos conjuntos de testes está apresentado na Tabela 11. Como descrito no capítulo de metodologia, os conjuntos de teste foram separados antes da compilação do conjunto final, denominado SH neste trabalho.

Tabela 11 – Cruzamento de treino e teste do YOLO12s em diferentes *datasets* de uso de capacete.

Conjunto de treino	Medidas no conjunto de teste (P; R; mAP50)*		
	SH	SFCHD	SHEL5K
SH	**0.96**;**0,90**;**0,96**	0,91;**0,86**;**0,92**	**0,92**;**0,85**;**0,93**
SFCHD	0,88;0,71;0,77	**0,94**;0,85;**0,92**	0,56;0,27;0,33
SHEL5K	0,69;0,48;0,52	0,45;0,38;0,34	0,90;0,85;0,90

*P: Precisão ; R: Revocação; mAP@0.5: Média da Precisão Média com *IoU* de 50%.

Os testes no próprio subconjunto de cada *dataset* mostraram um mAP acima de 90% e pertinente com outros trabalhos na área. Mesmo unindo várias imagens e utilizando imagens de cenários com baixa iluminação e com cenários com maior distância, ou objetos sobrepostos, como presentes nas imagens do conjunto SFCHD, o treinamento com o conjunto SH resultou em um mAP de 0,96, que é um valor alto para este tipo de contexto de detecção, como descrito no Capítulo 3.1.5 .

5.1.1 Comparação nos vídeos

Para melhorar a análise de desempenho de detecção dos modelos, foram processados os vídeos 1, 2, 3 e 4 (Capítulo 4.4) com cada um dos três modelos gerados. Dessa forma, pode-se inferir com maior segurança sobre o desempenho dos modelos em cenários externos distintos.

Para o vídeo 1 os valores de acertos e erros foram organizados em matrizes que permitem identificar os casos de detecções corretas ou incorretas, apresentadas na Figura 18. Essas matrizes sintetizam o eficácia dos modelos ao longo dos quadros do vídeo.

Baseando-se no total de ocorrências registradas em cada matriz, é possível calcular o percentual de 89,33% de acertos nas inferências realizadas para o treinamento com o SH. O mesmo cálculo foi feito para o SFCHD e o SHEL5K, revelando percentuais de 48,54% e 78,27%, respectivamente.

O modelo SFCHD apresentou um número elevado de falhas de classes e os exemplos dessas falhas de detecção pode ser vistos na Figura19. Os modelos SHEL5K e SH exibiram menos falhas de classes, entretando o SHEL5K mostrou maior dificuldade em identificar corretamente os objetos na imagem, com 39 não detecções. Já o modelo SH cometeu erros ao identificar o não uso de capacete em objetos que não eram relativos a região da cabeça da pessoa em 12 ocasiões. Alguns casos de falhas do SHEL5K e do SH estão também demonstrados na Figura19.

Para ampliar a investigação, foi escolhido o vídeo 2 para as comparações em um cenário distinto do vídeo 1. No vídeo 2, o ângulo de visão da câmera é elevado em relação ao solo, capturando imagens de forma semelhante a câmeras de vigilância. O indivíduo caminha no cenário utilizando e retirando o capacete.

As matrizes da Figura 20 mostram que, neste cenário, o modelo treinado com o *dataset* SH obteve desempenho superior: do total de ocorrências registradas em cada matriz, observa-se um percentual de 95,52

É relevante observar o erro de detecção dos três modelos na inferência do uso de capacete nos momentos em que o indivíduo passa em frente à abóbada, formando uma semelhança com o uso de capacete, podendo ser visto pela Figura 21 . Outras situações dinâmicas como essa podem ocorrer em cenários reais, assim ocasionando detecções incorretas.

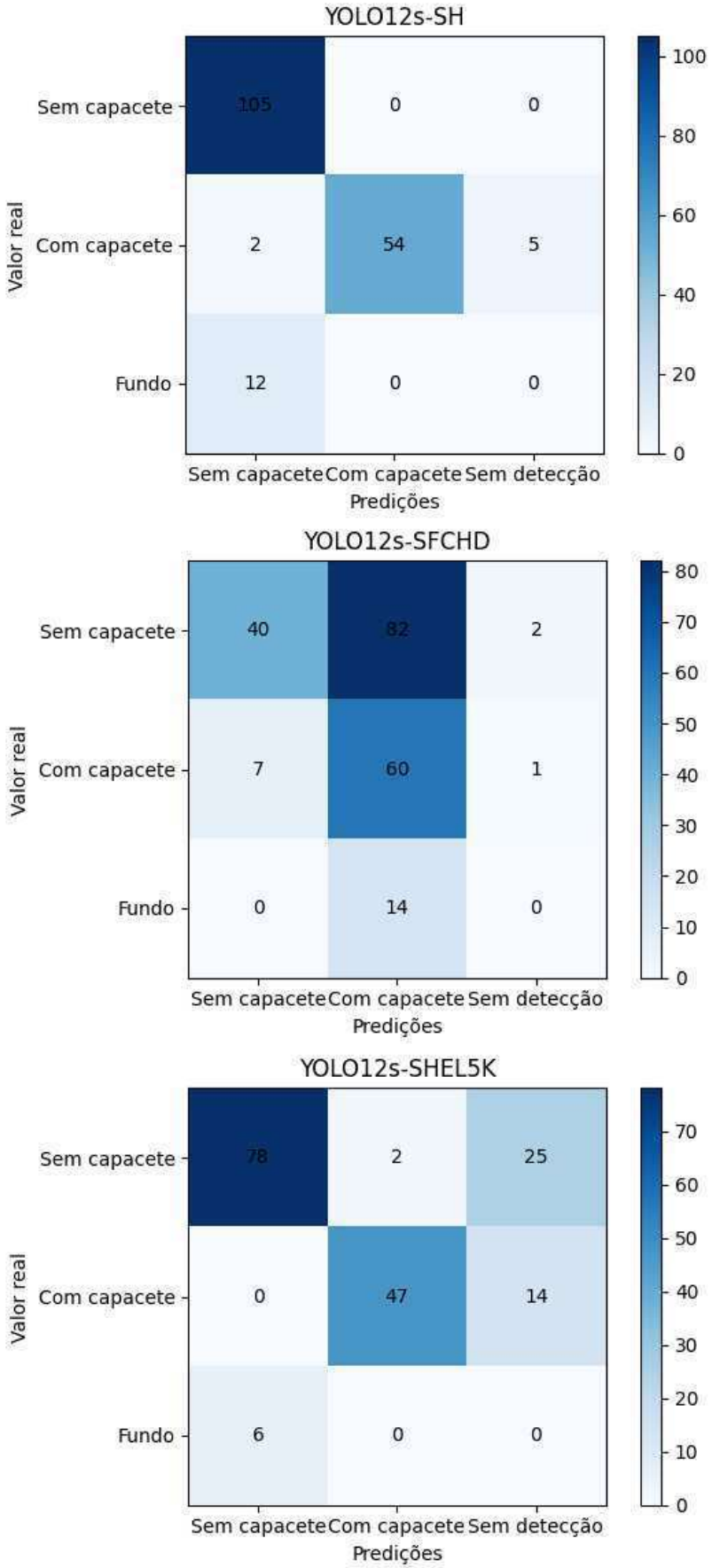


Figura 18 – Vídeo 1: Cruzamento de acertos e erros de detecções a partir dos modelos YOLO12s treinados nos *datasets* SH, SHEL5K e SFCHD.

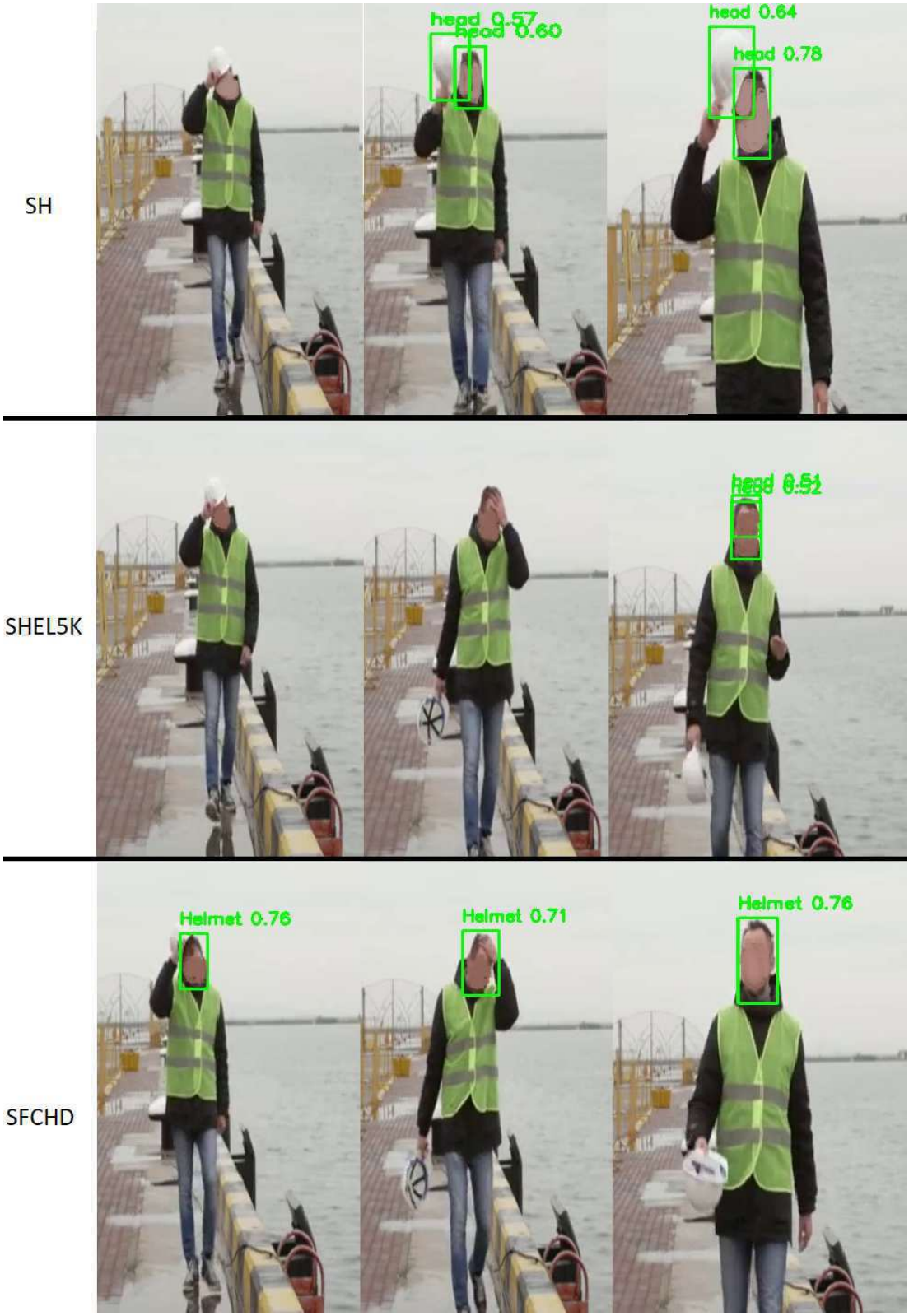


Figura 19 – Quadros de imagem do vídeo 1. Imagens em situações de falhas de detecções a partir dos modelos YOLO12s treinados nos *datasets* SH, SHEL5K e SFCHD. O rosto do indivíduo foi borrado para sua não identificação.

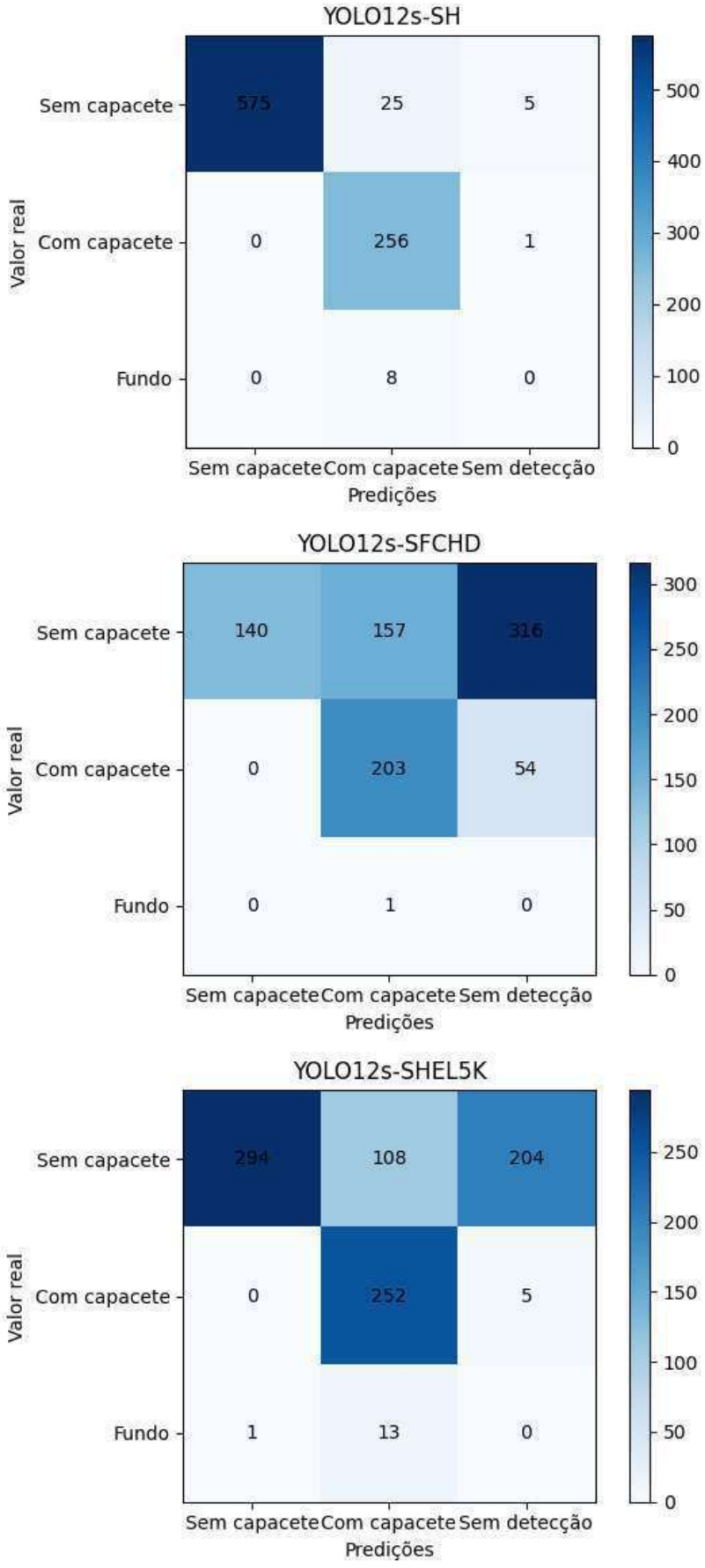


Figura 20 – Vídeo 2: Cruzamento de acertos e erros de detecções a partir dos modelos YOLO12s treinados nos *datasets* SH, SHEL5K e SFCHD.

Figura 21 – Quadros de imagem do vídeo 2. Imagens em situações de falhas de detecções a partir dos modelos YOLO12s treinados nos *datasets* SH, SHEL5K e SFCHD.

Outra situação utilizada para comparação foi o vídeo 3, que possui dois indivíduos utilizando capacetes de cores amarela e azul. A Figura 22 apresenta exemplos

de detecções para os três treinamentos.

Com os resultados das inferências dispostas pela Tabela 12, é possível verificar que os treinamentos com os *datasets* SH e SHEL5K foram eficazes na detecção de capacetes de cores distintas, enquanto o SFCHD não apresentou boa eficácia para treinar a rede na detecção de capacetes dessas cores, resultando em taxas elevadas de erro e desempenho impraticável para aplicações de automação.

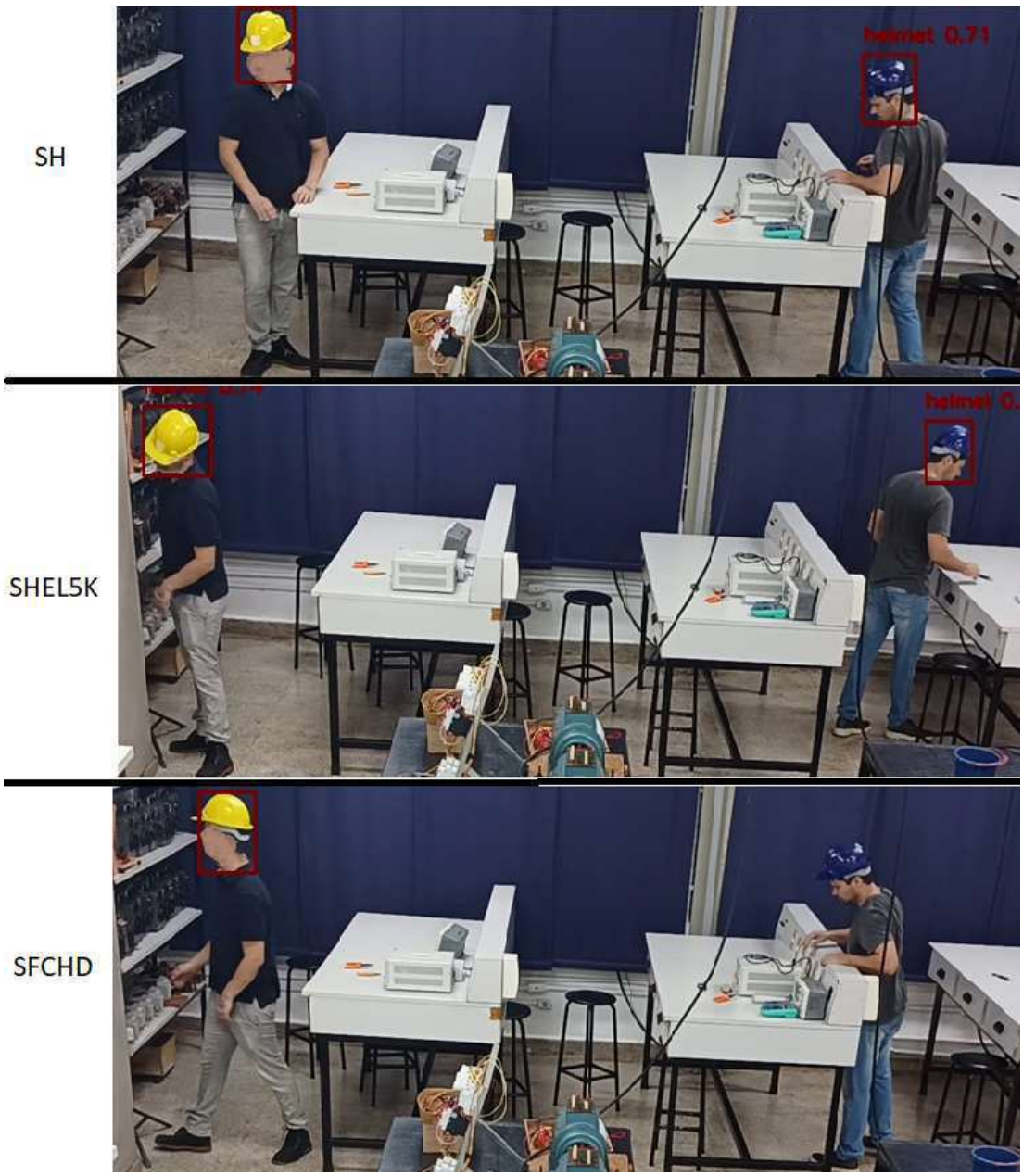


Figura 22 – Quadros de imagem do vídeo 3. Exemplo de detecções com capacetes de cores amarelo e azul a partir dos modelos YOLO12s treinados nos *datasets* SH, SHEL5K e SFCHD.

Tabela 12 – Detecções no vídeo 3. Vídeo com utilização de capacetes de cores amarelo e azul

Ocorrência	**SH**	**SHEL5K**	**SFCHD**
Detecções corretas: uso de capacete	558	568	248
Detecções incorretas: uso de capacete	0	0	11
Detecções incorretas: sem uso capacete	1	0	0
Sem detectar	24	15	318
Percentual de acerto	95,71%	97,43%	42,98%

Também foi realizada a comparação do desepenho dos modelos treinados no vídeo 4, que foi gravado em um ambiente com luminosidade inferior aos demais vídeos. A Figura 23 mostra as principais falhas ocorridas neste cenário, onde o SFCHD teve um desempenho ineficaz para realizar automações, enquanto os modelos SH e SHEL5K tiveram melhores performances, porém o SHEL5K teve um elevado número de detecções de objetos inexistentes. As matrizes da Figura 24 expõem as informações sobre as detecções dos três treinamentos. A partir dessas informações os percentuais de acertos do SH, SHEL5K e SFCHD foram respectivamente: 100%, 78% e 78,5%.

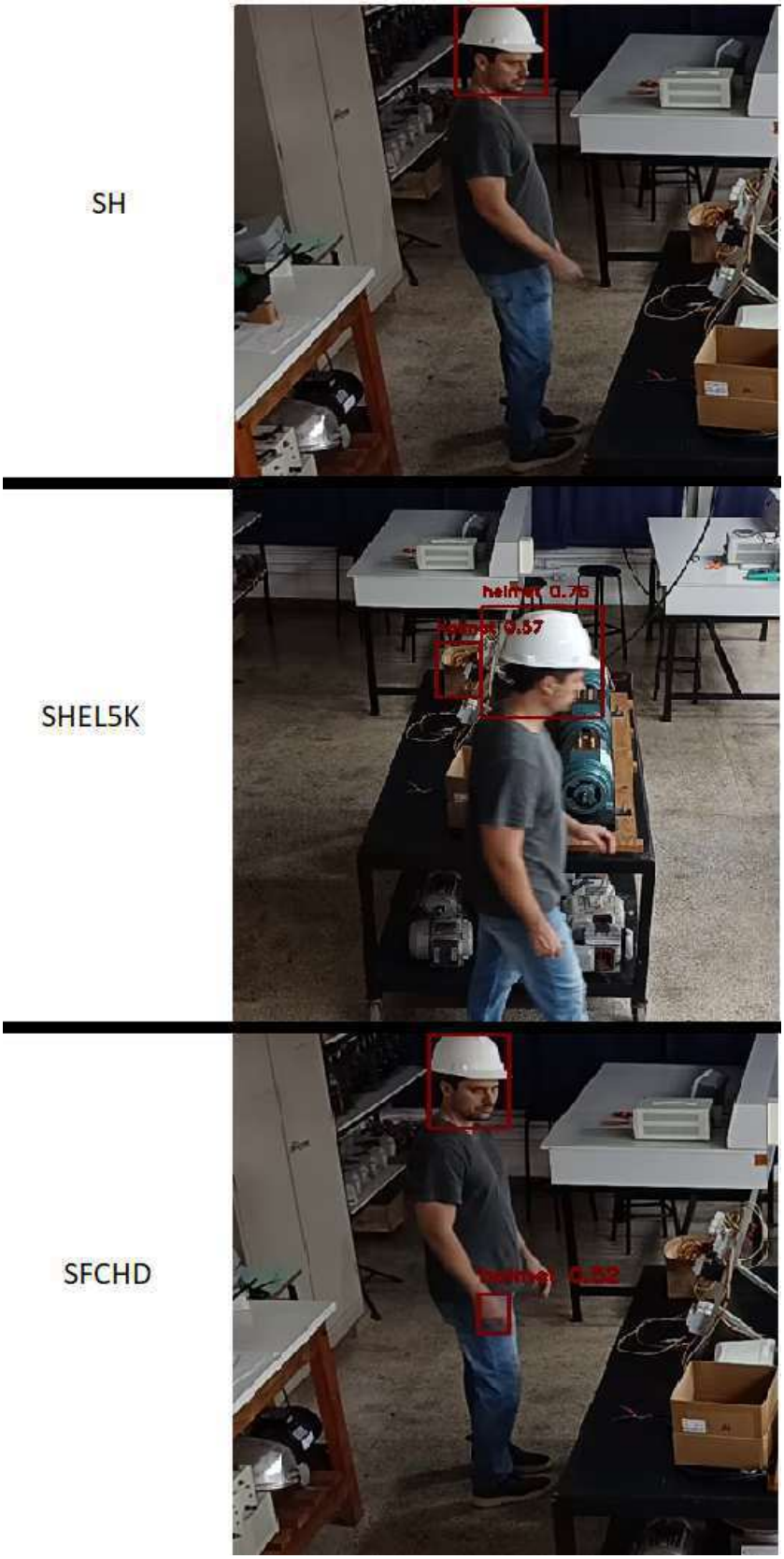


Figura 23 – Quadros de imagem do vídeo 4. Exemplo de detecções com capacetes em cenários com menor luminosidade a partir dos modelos YOLO12s treinados nos *datasets* SH, SHEL5K e SFCHD.

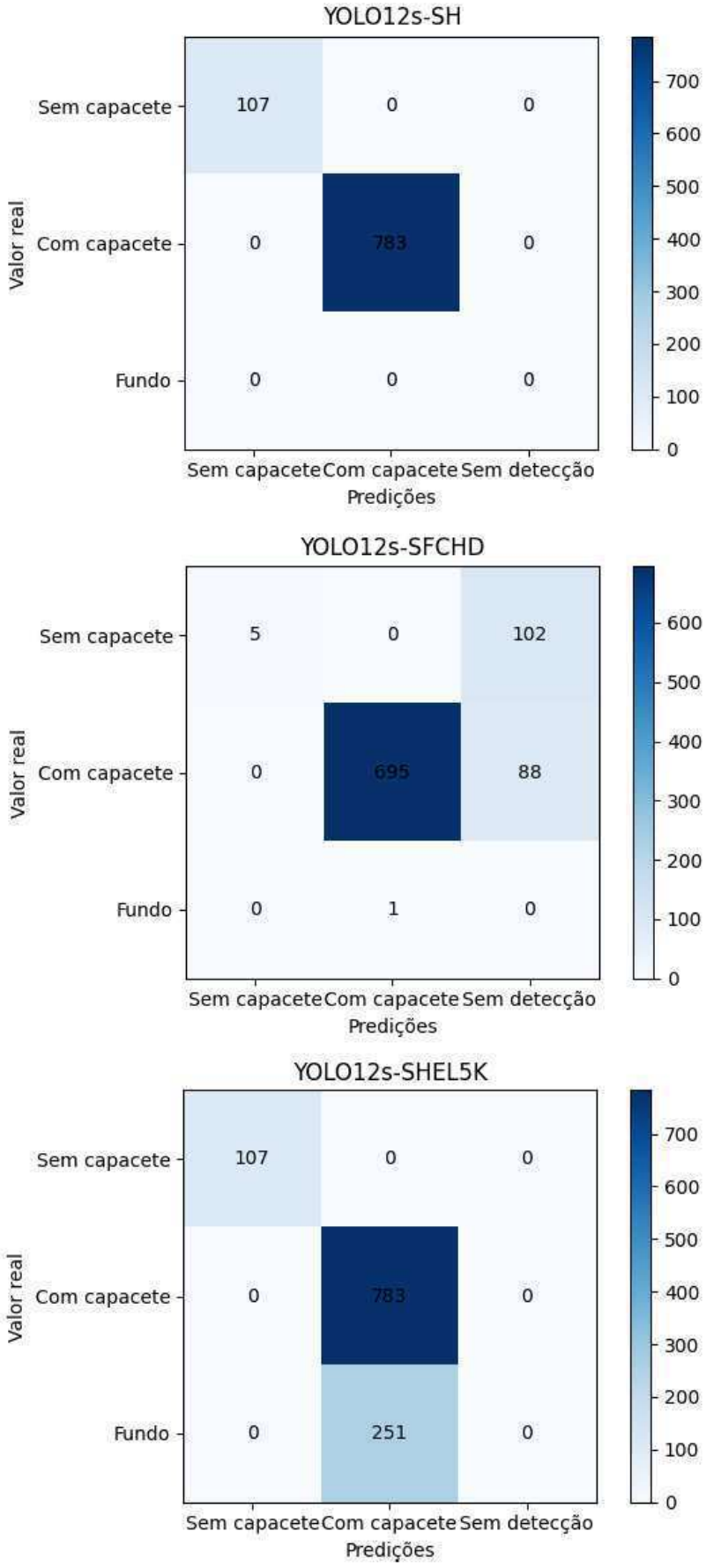


Figura 24 – Vídeo 4 com cruzamento de acertos e erros de detecções a partir dos modelos YOLO12s treinados nos *datasets* SH, SHEL5K e SFCHD.

Pelas verificações de desempenho dos modelos, o treinamento com o *dataset* SH mostrou-se mais eficaz para a aplicação em cenários reais e para a finalidade de testes de aplicações em sistemas autônomos. Dessa forma, a seguir, são investigadas as detecções com o modelo YOLO *small* com *dataset* SH, YOLO12s-SH.

5.2 Testes com o YOLO12s-SH

O modelo YOLO treinado com o conjunto de imagens definitivo passou por mais um teste relevante através do vídeo 5 que possui um indivíduo caminhando utilizando boné. Pela Figura 25, podem ser vistos exemplos de detecção com acertos em disposições distintas da pessoa e um erro de detecção, confundindo o boné com a utilização de capacete.

A contabilização dos acertos e erros de detecção está disposta na Tabela 13, que demonstra que a maioria das detecções foi correta, porém com um número elevado de falsos registros de uso de capacete, com percentual de 18,63% em relação ao total. Esse indicador de funcionamento do detector não impediria sua implementação em um sistema de automação, devido ao elevado percentual de acertos na ocorrência de não uso do EPI. Dessa forma, caso o indivíduo utilize boné em um local de inspeção, serão detectadas várias ocorrências de não uso do EPI, e o pós-processamento dessas informações pode acionar o alarme com precisão.

Figura 25 – Vídeo 5 com exemplos de utilização de boné. Quadros processados pelo detector YOLO12s-SH.

Tabela 13 – Detecções no vídeo 5 com o detector YOLO12s-SH. Vídeo com utilização de boné.

Ocorrência	**Quantidade**
Detecções corretas: não uso de capacete	691
Detecções incorretas: uso de capacete	168
Sem detectar	43
Percentual de acerto	76,61%

Em outra análise, o impacto do distanciamento entre o indivíduo e a câmera sobre o desempenho das detecções foi avaliado com base nos valores de confiança das detecções.

Para cada distância analisada, foram calculadas as médias de 150 detecções realizadas para os casos: com uso de capacete e sem o uso do equipamento. Os resultados obtidos são apresentados na Tabela 14 que mostra o decréscimo de confiança das detecções conforme a pessoa se afasta da câmera. Foi notado que na transição para 12 metros ocorreu um descréscimo mais expressivo na qualidade das inferências.

Pelas observações, na distância de 15 metros as detecções se aproximaram do limiar de 0,50 e em muitos casos ocorreram falsas detecções.

Avaliando um cenário de operação deste detector, a distância de 12 metros, neste contexto da Figura 16, mostra-se como uma distância limite para aplicar os processamentos posteriores a fim de filtrar e extrair a informação se o indivíduo utiliza ou não o EPI.

Tabela 14 – Resultados médios de confiança das detecções em função da distância

Distância (m)	Confiança média (com capacete)	Confiança média (sem capacete)
3	0,8169	0,8180
6	0,7874	0,7979
9	0,7703	0,7172
12	0,6251	0,6603
15	0,5993	0,6323

5.3 Algoritmo de filtragem aplicados

Os algoritmos de filtragem de erros foram implementados com o objetivo de avaliar seu desempenho em um sistema que utiliza vídeos como entrada. A seguir, são apresentados os resultados obtidos nas etapas de testes dos diferentes modos de filtragem: sem o uso de rastreadores, com rastreadores e com rastreadores associados à reidentificação.

5.3.1 Filtragens aplicadas ao vídeo 1

Como descrito nas seções anteriores, o vídeo 1, apresentado na Figura 19, apresenta alguns erros de detecção, incluindo falsas detecções de não uso de capacete. Para averiguar a eficiência da filtragem desses erros, foram aplicados todos os métodos de filtragem considerados neste trabalho.

A partir da Figura 26, verifica-se que, com a aplicação das filtragens, ocorreu uma redução no número de alarmes, principalmente devido à eliminação de detecções

de não uso de capacete fora da área de correspondência, situação observada no caso sem filtro aplicado.

Nesse vídeo, os quatro métodos de filtragem apresentaram alarmes pertinentes ao não uso de EPI e, mesmo com a redução da taxa de amostragem de quadros por segundo, ocorreu ao menos um alarme na área de não uso de capacete.

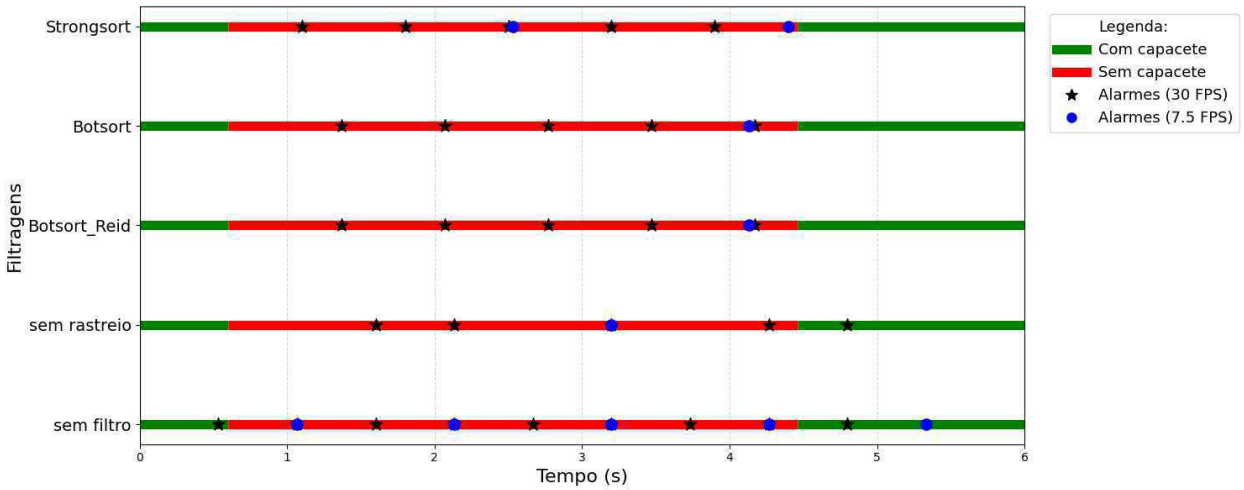


Figura 26 – Vídeo 1: geração de alarmes a partir dos meios de filtragem aplicados. Quadros processados pelo detector YOLO12s-SH.

5.3.2 Filtragens aplicadas ao vídeo 6

Como descrito na metodologia deste documento, o vídeo 6, com imagens presentes na Figura 27, contém dois indivíduos caminhando no cenário. Para averiguar o funcionamento do rastreador em cenários com mais de uma pessoa, foram aplicados todos os métodos de filtragem considerados neste trabalho.

Figura 27 – Vídeo 6: imagens de resultado do rastreador Bot-SORT aplicado. Quadros processados pelo detector YOLO12s-SH.

À medida que ocorreram oscilações nas detecções, principalmente em função da retirada e recolocação do capacete, os identificadores foram sendo alterados e, a partir deles, as filtragens dos métodos com rastreio foram realizadas. Dessa forma, quando ocorre a repetição de um identificador persistente associado à classe 0, há uma maior probabilidade de que realmente tenha ocorrido o não uso de EPI por aquele

indivíduo.

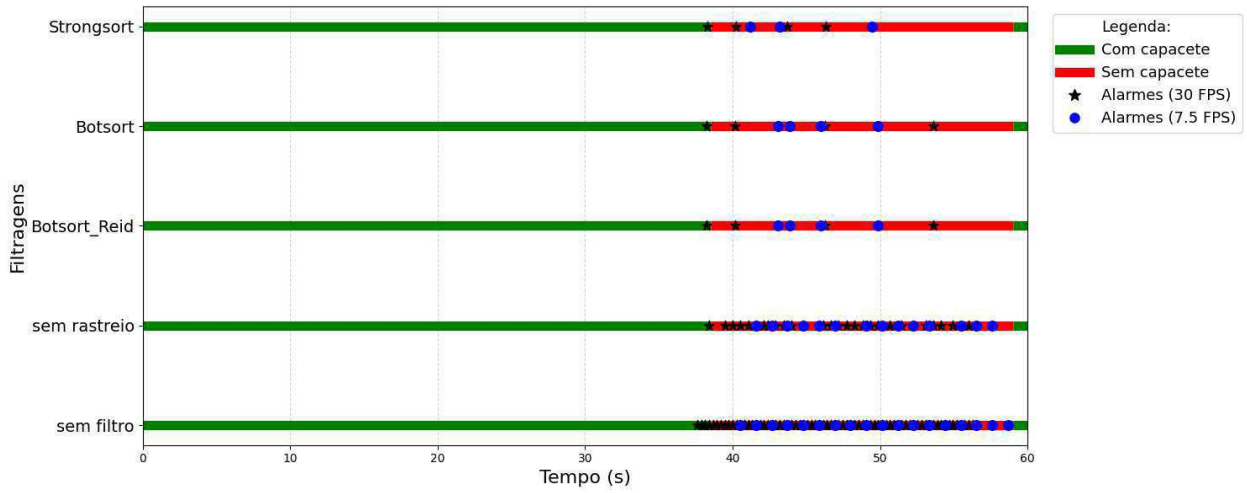


Figura 28 – Vídeo 6: geração de alarmes a partir dos meios de filtragem aplicados. Quadros processados pelo detector YOLO12s-SH.

A partir da Figura 28, verifica-se que, com a aplicação das filtragens, ocorreu uma redução no número de alarmes, que foi pequena e proporcional ao tempo de vídeo amostrado. Também não foram identificadas alterações nos resultados ao se comparar o uso ou não da etapa de reidentificação no Bot-SORT, controlada pelos parâmetros *appearance thresh* e *with reid*.

5.3.3 Filtragens aplicadas ao vídeo 7

Como descrito na metodologia deste documento, o vídeo 7, com imagens presentes na Figura 29, contém dois indivíduos caminhando no cenário, se deslocando por toda a sala. Os rastreadores foram implementados e mostraram a criação de identificadores que persistiram no indivíduo até a troca de classe pelo detector.

Figura 29 – Vídeo 7: imagens de resultado do rastreador Bot-SORT aplicado. Quadros processados pelo detector YOLO12s-SH.

Neste vídeo ocorreram duas ocasiões de não uso de capacete, que estão representados na Figura 30. Nas duas ocasiões ocorreram gatilhos de ativação de alarmes e também um diminuição dos alarmes gerados comparativamente ao caso sem filtro e sem rastreador.

Ainda neste vídeo notou-se a não variação de resultados pela remoção da reidentificação do BotSORT.

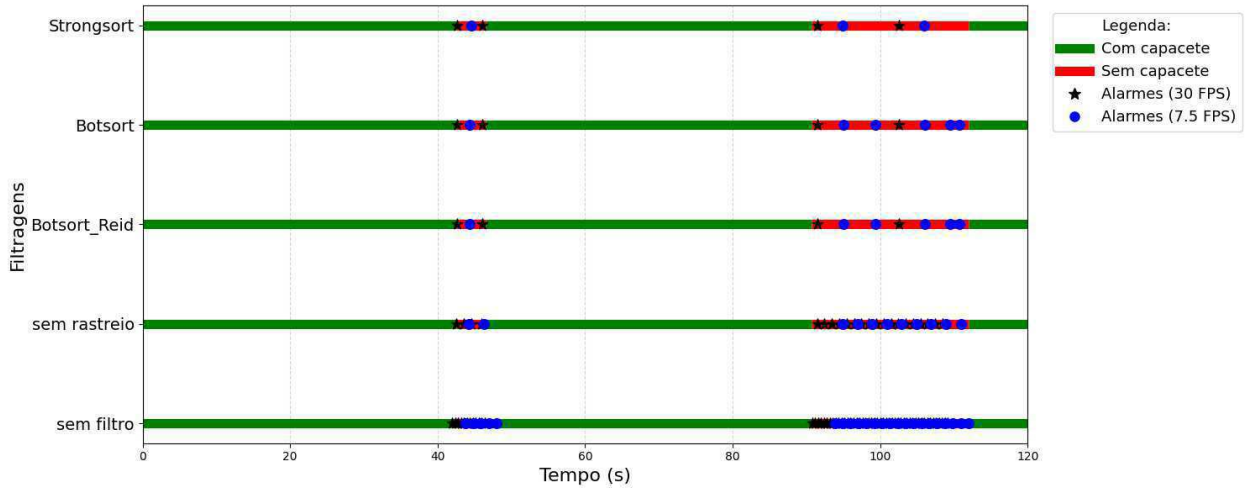


Figura 30 – Vídeo 7: geração de alarmes a partir dos meios de filtragem aplicados. Quadros processados pelo detector YOLO12s-SH.

Como não ocorreram detecções errôneas neste vídeo, mesmo sem filtragem os alarmes ficaram correspondentes a área de não uso de EPI. Porém, ficou evidente a diminuição de número de alarmes pelos outros métodos.

5.3.4 Filtragens aplicadas ao vídeo 8

O vídeo 8, com imagens apresentadas na Figura 31, teve, em seus testes de filtragem, a inserção de erros. Os resultados mostraram que, quando não ocorre o preenchimento do gatilho, os alarmes não são disparados. No entanto, ao aplicar quantidades de 30 ou 40 quadros errôneos, os alarmes são disparados e o rastreador se mantém com o mesmo ID. Dessa forma, também foi verificada a não repetição de alarmes gerados pelo mesmo identificador, situação que poderia ocorrer em casos de erros persistentes em um cenário.

Figura 31 – Exemplos de captura de imagem e detecção no vídeo 8. Em uma cena, o indivíduo sai parcialmente do quadro de imagem e, em outra, inclina-se, deixando o capacete na horizontal.

Outra situação percebida neste vídeo foi a ocorrência de falso positivo para o não uso do capacete em alguns quadros, nos momentos em que o indivíduo sai parcialmente do quadro de imagem, permanecendo com parte da cabeça visível. Nessa condição, o detector realizou detecções errôneas; porém, pelas filtragens com os rastreadores, como mostra a Figura 32, esse alarme não foi disparado. Ainda assim, esse é um caso que pode ocorrer em cenários reais, e a definição de faixas de detecção válidas pode prevenir esse tipo de erro.

Quanto às filtragens realizadas, notou-se a diminuição do número de alarmes e também a semelhança de resultados quanto ao uso ou não da reidentificação.

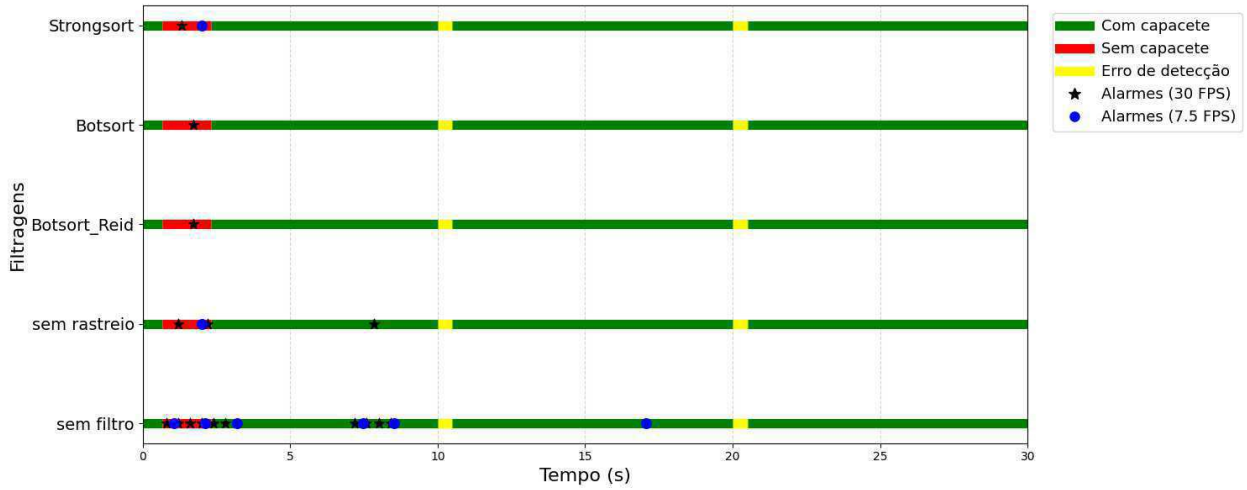


Figura 32 – Vídeo 8: geração de alarmes a partir dos meios de filtragem aplicados. Quadros processados pelo detector YOLO12s-SH.

5.3.5 Filtragens aplicadas ao vídeo 9

Os testes realizados no vídeo 9 demonstraram o rastreio do indivíduo ao longo do caminho percorrido por ele no cenário. Na Figura 33, é apresentado o caso de fixação do identificador quando ocorrem oscilações do detector. No quadro 541, o detector não envia detecções e, ainda assim, o identificador se manteve associado ao mesmo indivíduo. Essa quantidade de quadros em que o rastreador permite permanecer sem detecções é configurada pelos parâmetros de cada rastreador. Os parâmetros utilizados foram testados ao longo dos diversos ensaios em vídeo e estão descritos no Capítulo de Metodologia.

Figura 33 – Vídeo 9: imagens de resultado do rastreador Strong-SORT aplicado. Quadros processados pelo detector YOLO12s-SH.

Neste vídeo, as detecções apresentam oscilações quando o indivíduo se en-

contra mais distante, a cerca de 12 metros da câmera. Nesse contexto, os rastreadores tiveram a função de manter o rastro da detecção e possibilitaram o disparo de alarmes após o atingimento do limiar estabelecido. A Figura 34 apresenta os alarmes gerados por todos os métodos de filtragem e também evidencia a diminuição do número de alarmes quando comparado ao método sem rastreio ou sem filtragem.

Observando a filtragem com 7,5 FPS, nota-se que os rastreadores disparam significativamente menos alarmes, sendo apenas um para o Bot-SORT e dois para o Strong-SORT. Essa quantidade pode ser configurada conforme os requisitos de implementação em software para os cenários. No caso de se buscar maior precisão no disparo e maior confiabilidade, o aumento da janela deslizante com o uso de 7,5 FPS possibilita que os alarmes ocorram apenas em situações em que o indivíduo permaneça por um intervalo de tempo maior, com múltiplas confirmações do não uso de EPI.

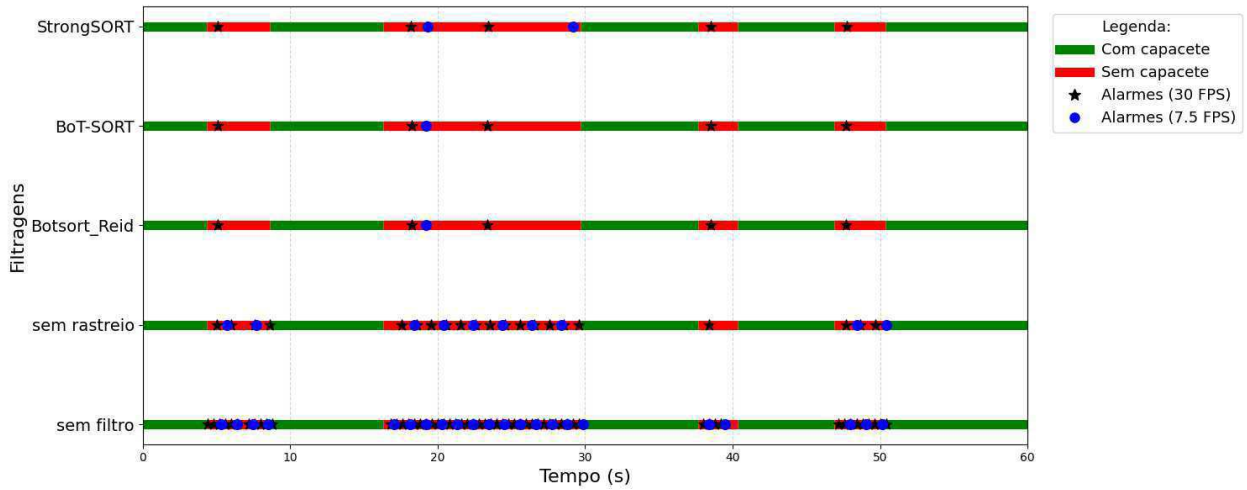


Figura 34 – Vídeo 9: geração de alarmes a partir dos meios de filtragem aplicados. Quadros processados pelo detector YOLO12s-SH.

5.3.6 Comparação da velocidade de execução

A avaliação do tempo para processamento computacional para cada algoritmo de filtragem e rastreamento, foi realizada por meio da comparação do número médio de quadros por segundo. Os quatro métodos de associação temporal: filtragem sem rastreamento, BotSORT sem ReID, StrongSORT e BotSORT com ReID foram implementados neste teste. Todos os experimentos foram conduzidos utilizando o mesmo vídeo de entrada (vídeo 8, descrito na metodologia) e o mesmo modelo de detecção YOLO-12s-SH, variando-se exclusivamente o mecanismo de associação temporal empregado.

A Figura 35 apresenta a comparação das velocidades de processamento obtidas para cada método. Observa-se que a abordagem sem rastreamento apresentou

a maior agilidade para ser processada, alcançando 60,08 FPS.

A utilização do BotSORT sem o módulo de identificação por CNN (ReID) resultou em uma redução moderada da velocidade de processamento, com FPS de 47,95. Ainda assim, esse valor se mantém elevado, indicando que o custo associado ao rastreamento baseado predominantemente em informações geométricas e de movimento é relativamente baixo e compatível com aplicações em tempo real.

Os casos que incorporam a reidentificação apresentaram um FPS médio de 27,96 para o StrongSORT, enquanto o BotSORT com ReID apresentou desempenho semelhante, com 27,73 FPS. Esses resultados evidenciam o impacto computacional da etapa de ReID, que demanda inferência adicional por CNN para cada detecção.

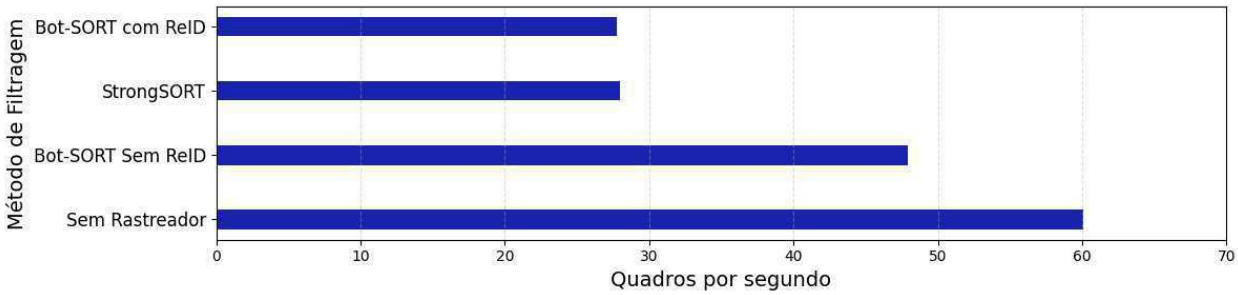


Figura 35 – Comparação da velocidade de processamento dos vídeos pelos algoritmos de filtragem

5.4 Contribuições

Diante dos questionamentos apresentados no Capítulo 2.2.1, elencam-se as principais contribuições deste trabalho:

1. **Investigação da capacidade de generalização em vídeos externos aos conjuntos de treinamento.** Foi avaliado o desempenho dos modelos YOLO aplicados a vídeos reais não pertencentes aos contextos de treinamento.

2. **Avaliação da necessidade de rastreadores em sistemas de detecção de EPI.** Investigou-se se o uso de rastreadores, como o StrongSORT, que incorpora uma CNN para associação de identidades e impõe elevado custo computacional, é realmente necessário, por meio da comparação com alternativas computacionalmente menos onerosas.

3. **Comparação entre StrongSORT e BotSORT sem reidentificação.** Foram implementados os algoritmos StrongSORT e BotSORT, incluindo a avaliação do BotSORT com e sem a etapa de identificação por CNN, possibilitando mensurar os efeitos de cada abordagem no contexto da detecção do uso de capacete.

4. **Aplicação e análise do método de filtragem temporal sem rastreador.** O método proposto por Zaidi2024, baseado no uso de buffers e na consistência temporal das detecções, foi aplicado como alternativa ao rastreamento e filtragem por identificadores. A análise permitiu avaliar sua viabilidade na mitigação de falsos positivos em vídeos de cenários externos aos dos *datasets* de treinamento.

6 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um sistema de visão computacional para inspeção automática do uso de capacetes de segurança, abordando desde a compilação de um *dataset* capaz de atingir métricas pertinentes com o estado da arte até a implementação de técnicas de filtragem temporal e rastreamento de objetos. Os resultados demonstraram a eficácia do modelo YOLOv12s treinado no conjunto SH. A combinação entre a detecção e os algoritmos de rastreamento mostrou-se viável para reduzir falsos alarmes, garantindo que os alarmes sejam acionados apenas em casos consistentes de não conformidade.

A análise comparativa entre as abordagens de filtragem temporal e de rastreamento, em relação aos casos sem rastreamento e aos métodos BoT-SORT, com e sem reidentificação, e StrongSORT, evidenciou o impacto da etapa de reidentificação no desempenho computacional do sistema. Observou-se que a utilização de mecanismos de reidentificação acarreta um custo significativo no tempo de processamento.

A filtragem temporal sem o uso de rastreadores mostrou-se eficaz nos cenários avaliados, contudo, verificou-se um aumento no número de alarmes gerados quando comparada às abordagens baseadas em rastreamento. Esse comportamento decorre da presença de múltiplas detecções independentes ao longo dos quadros, incluindo falsas detecções, que influenciam diretamente o acionamento dos alarmes.

Em relação à redução da taxa de quadros por segundo, constatou-se que os alarmes continuaram sendo corretamente acionados e que os processos de filtragem permaneceram funcionais, desde que fossem realizados os devidos ajustes nos parâmetros dos rastreadores. Esses resultados indicam a robustez das estratégias de filtragem e rastreamento mesmo em condições de menor frequência temporal.

A aplicação da menor taxa de quadros por segundo com uso do rastreador Bot-SORT sem reidentificação se mostrou eficaz para a tarefa, realizando o tratamento dos vídeos com um percentual 71% mais ágil. Dessa forma, se mostrou como uma solução para implementações sem onerar demasiadamente o tempo de processamento.

Ainda, pelos trabalhos, os testes demonstram uma boa performace desses algoritmos em cenários de ambientes fechados, com boa iluminação e com alcance máximo de 12 metros.

6.1 Trabalhos Futuros

Como continuidade deste trabalho, destaca-se a relevância da implementação de protótipos para testes em campo, permanecendo por longos períodos de testes em ambientes industriais, nos quais os alarmes gerados possam ser analisados, bem como outros possíveis erros de detecção.

Nesse cenário de testes, uma opção viável seria a análise da incorporação de imagens do próprio ambiente a ser automatizado, reduzindo a probabilidade de falsas detecções associadas ao fundo da imagem. Essa etapa seria relevante para investigar o impacto real da incorporação dessas imagens nos resultados obtidos.

Também poderiam ser realizadas novas implementações em conjunto de imagens, realizando a estratificação dos cenários, como, por exemplo, por tipo de iluminação, distância e tipo de ambiente.

REFERÊNCIAS

ABBASIANJAHROMI, H.; GHAZVINI, E. S. Developing a Wearable Device Based on IoT to Monitor the Use of Personal Protective Equipment in Construction Projects. **Iran Journal of Science and Technology, Transactions of Civil Engineering**, v. 46, p. 2561–2573, 2022. DOI: https://doi.org/10.1007/s40996-021-00716-6.

AHARON, N.; ORFAIG, R.; BOBROVSKY, B.-Z. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. **arXiv preprint arXiv:2206.14651**, 2022. Disponível em: https://arxiv.org/abs/2206.14651.

AHMAD, H. M.; RAHIMI, A. SH17: A dataset for human safety and personal protective equipment detection in manufacturing industry. **Journal of Safety Science and Resilience**, 2024. DOI: https://doi.org/10.1016/j.jnlssr.2024.09.002.

ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a Convolutional Neural Network. *In:* PROCEEDINGS of the 2017 International Conference on Engineering and Technology (ICET). Antalya, Turkey: IEEE, 2018. p. 1–6. DOI: https://doi.org/10.1109/ICEngTechnol.2017.8308186.

ALI, M. L.; ZHANG, Z. The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection. **Computers**, v. 13, p. 336, 2024. DOI: https://doi.org/10.3390/computers13120336.

AN, Q.; XU, Y.; YU, J.; TANG, M.; LIU, T.; XU, F. Research on Safety Helmet Detection Algorithm Based on Improved YOLOv5s. **Sensors**, v. 23, n. 13, p. 5824, 2023. DOI: https://doi.org/10.3390/s23135824.

ANDREWMVD. **Hard Hat Detection Dataset**. Mountain View, USA, 2020. Disponível em: https://www.kaggle.com/datasets/andrewmvd/hard-hat-detection.

BALDISSONE, G.; COMBERTI, L.; BOSCA, S.; MURÈ, S. The analysis and management of unsafe acts and unsafe conditions: Data collection and analysis. **Safety Science**, v. 119, p. 240–251, 2019. DOI: https://doi.org/10.1016/j.ssci.2018.10.006.

BEWLEY, A.; GE, Z.; OTT, L.; RAMOS, F.; UPCROFT, B. Simple Online and Realtime Tracking. *In:* PROCEEDINGS of the 2016 IEEE International Conference on Image Processing (ICIP). Phoenix, AZ, USA: IEEE, 2016. p. 3464–3468. DOI: https://doi.org/10.1109/ICIP.2016.7533003.

BROSTRÖM, M. BoxMOT: A collection of SOTA real-time, multi-object trackers for object detectors. **Zenodo**, 2023. DOI: https://doi.org/10.5281/zenodo.8132989.

CLARO, M.; VOGADO, L.; SANTOS, J.; VERAS, R. Utilização de Técnicas de Data Augmentation em Imagens: Teoria e Prática. *In:* MINICURSOS da ERCEMAPI 2020. [*S. l.: s. n.*], 2020. DOI: https://doi.org/10.5753/sbc.11.5.3. Disponível em: https://books-sol.sbc.org.br/index.php/sbc/catalog/view/48/217/455.

DAVIES, E. R. **Machine Vision: Theory, Algorithms, Practicalities**. 3. ed. Waltham, USA: Pearson Prentice Hall, 2005.

DENDORFER, P.; OŠEP, A.; MILAN, A.; SCHINDLER, K.; CREMERS, D.; REID, I.; ROTH, S.; LEAL-TAIXÉ, L. MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking. **International Journal of Computer Vision**, v. 129, n. 4, p. 845–881, 2021. DOI: https://doi.org/10.1007/s11263-020-01393-0.

DI, B.; XIANG, L.; DAOQING, Y.; KAIMIN, P. MARA-YOLO: An Efficient Method for Multiclass Personal Protective Equipment Detection. **IEEE Access**, v. 12, p. 24866–24878, 2024. DOI: https://doi.org/10.1109/ACCESS.2024.3365504.

DU, Y.; ZHAO, Z.; SONG, Y.; ZHAO, Y.; SU, F.; GONG, T.; MENG, H. StrongSORT: Make DeepSORT Great Again. **arXiv preprint arXiv:2202.13514**, 2022. Disponível em: https://arxiv.org/abs/2202.13514.

DWYER, B.; NELSON, J.; HANSEN, T. **Roboflow**. Des Moines, IA, USA, 2024. Disponível em: https://roboflow.com/.

ELGENDY, M. **Deep Learning for Vision Systems**. Shelter Island, NY: Manning Publications, 2020.

FANG, Q.; LI, H.; LUO, X.; DING, L.; LUO, H.; ROSE, T. M.; AN, W. Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. **Automation in Construction**, v. 85, p. 1–9, 2018. DOI: https://doi.org/10.1016/j.autcon.2017.09.018.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *In:* PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Columbus, OH, USA: IEEE, 2014. p. 580–587. DOI: https://doi.org/10.1109/CVPR.2014.81.

GOCHOO, M. **Safety Helmet Wearing Dataset**. Amsterdam, Netherlands, 2021.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA, USA: MIT Press, 2016.

HU, L.; REN, J. YOLO-LHD: An enhanced lightweight approach for helmet wearing detection in industrial environments. **Frontiers in Built Environment**, v. 9, p. 1288445, 2023. DOI: https://doi.org/10.3389/fbuil.2023.1288445.

ILO. **Nine Business Practices for Improving Safety and Health through Supply Chains and Building a Culture of Prevention and Protection**. Geneva, 2021. Disponível em: https://www.ilo.org/sites/default/files/wcmsp5/groups/public/%40dgreports/%40dcomm/documents/briefingnote/wcms_821481.pdf.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Journal of Basic Engineering**, v. 82, n. 1, p. 35–45, 1960. DOI: https://doi.org/10.1115/1.3662552.

KE, X.; CHEN, W.; GUO, W. 100+ FPS detector of personal protective equipment for worker safety: A deep learning approach for green edge computing. **Peer-to-Peer Networking and Applications**, v. 15, p. 950–972, 2022. DOI: https://doi.org/10.1007/s12083-021-01258-4.

KELM, A.; LAUSSAT, L.; MEINS-BECKER, A.; PLATZ, D.; KHAZAEE, M. J.; COSTIN, A. M.; HELMUS, M.; TEIZER, J. Mobile passive Radio Frequency Identification (RFID) portal for automated and rapid control of Personal Protective Equipment (PPE) on construction sites. **Automation in Construction**, v. 36, p. 38–52, 2013. DOI: https://doi.org/10.1016/j.autcon.2013.08.017.

KIM, J. H.; JO, B. W.; JO, J. H.; LEE, Y. S.; KIM, D. K. Autonomous Detection System for Non-Hard-Hat Use at Construction Sites Using Sensor Technology. **Sustainability**, v. 13, n. 3, p. 1102, 2021. DOI: https://doi.org/10.3390/su13031102.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. **arXiv preprint arXiv:1412.6980**, 2015.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. *In:* ADVANCES in Neural Information Processing Systems (NeurIPS). Lake Tahoe, NV, USA: Curran Associates, Inc., 2012. p. 1097–1105.

KUHN, H. W. The Hungarian Method for the assignment problem. **Naval Research Logistics Quarterly**, v. 2, n. 1-2, p. 83–97, 1955. DOI: https://doi.org/10.1002/nav.3800020109.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, p. 436–444, 2015.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998. DOI: https://doi.org/10.1109/5.726791.

LEE, J.-Y.; CHOI, W.-S.; CHOI, S.-H. Verification and Performance Comparison of CNN-Based Algorithms for Two-Step Helmet-Wearing Detection. **Expert Systems**

with Applications, v. 225, p. 120096, 2023. DOI: https://doi.org/10.1016/j.eswa.2023.120096.

LEE, S.-K.; YU, J.-H. Ontological inference process using AI-based object recognition for hazard awareness in construction sites. **Automation in Construction**, v. 153, p. 104961, 2023. DOI: https://doi.org/10.1016/j.autcon.2023.104961.

LEMA, D. G.; USAMENTIAGA, R.; GARCÍA, D. F. Low-cost system for real-time verification of personal protective equipment in industrial facilities using edge computing devices. **Journal of Real-Time Image Processing**, v. 20, p. 111, 2023. DOI: https://doi.org/10.1007/s11554-023-01368-7.

LIN, B. Safety Helmet Detection Based on Improved YOLOv8. **IEEE Access**, v. 12, p. 28260–28272, 2024. DOI: https://doi.org/10.1109/ACCESS.2024.3368161.

LIN, T.-Y.; DOLLÁR, P.; GIRSHICK, R.; HE, K.; HARIHARAN, B.; BELONGIE, S. Feature Pyramid Networks for Object Detection. *In:* PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA: IEEE, 2017. p. 936–944. DOI: https://doi.org/10.1109/CVPR.2017.106.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft COCO: Common Objects in Context. *In:* PROCEEDINGS of the European Conference on Computer Vision (ECCV). Zurich, Switzerland: Springer, 2014. p. 740–755.

LITWILLER, D. CCD vs. CMOS: Facts and Fiction. **Photonics Spectra**, p. 154–158, 2005.

LIU, S.; QI, L.; QIN, H.; SHI, J.; JIA, J. Path Aggregation Network for Instance Segmentation. *In:* PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City, UT, USA: IEEE, 2018. p. 8759–8768. DOI: https://doi.org/10.1109/CVPR.2018.00913.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A. C. SSD: Single Shot MultiBox Detector. *In:* LECTURE Notes in Computer Science. [*S. l.*]: Springer, Cham, 2016. v. 9905, p. 21–37.

LO, J.-H.; LIN, L.-K.; HUNG, C.-C. Real-Time Personal Protective Equipment Compliance Detection Based on Deep Learning Algorithm. **Sustainability**, v. 15, p. 391, 2023. DOI: https://doi.org/10.3390/su15010391.

LOSHCHILOV, I.; HUTTER, F. Decoupled Weight Decay Regularization. *In:* PROCEEDINGS of the International Conference on Learning Representations (ICLR). New Orleans, LA, USA: ICLR, 2019. Disponível em: https://arxiv.org/abs/1711.05101.

MAHALANOBIS, P. C. On the generalised distance in statistics. **Proceedings of the National Institute of Sciences of India**, v. 2, p. 49–55, 1936.

MPS. **Anuário Estatístico de Acidentes de Trabalho: AEAT 2023**. Brasília, DF, 2023. Disponível em: https://www.gov.br/previdencia/pt-br/assuntos/previdencia-social/arquivos/AEAT-2023/secao-i-estatisticas-de-acidentes-do-trabalho/subsecao-a-acidentes-do-trabalho/capitulo-1-brasil-e-grandes-regioes.

NATH, N. D.; BEHZADAN, A. H.; PAAL, S. G. Deep learning for site safety: Real-time detection of personal protective equipment. **Automation in Construction**, v. 112, p. 103085, 2020. DOI: https://doi.org/10.1016/j.autcon.2020.103085.

OTGONBOLD, M.-E.; GOCHOO, M.; ALNAJJAR, F.; ALI, L.; TAN, T.-H.; HSIEH, J.-W.; CHEN, P.-Y. SHEL5K: An Extended Dataset and Benchmarking for Safety Helmet Detection. **Sensors**, v. 22, n. 6, p. 2315, 2022. DOI: https://doi.org/10.3390/s22062315.

PENG, D.; SUN, Z.; CHEN, Z.; CAI, Z.; XIE, L.; JIN, L. Detecting Heads using Feature Refine Net and Cascaded Multi-scale Architecture. **arXiv preprint arXiv:1803.09256**, 2018.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 6, p. 779–788, 2016. DOI: https://doi.org/10.1109/TPAMI.2016.2577031.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 6, p. 1137–1149, 2017. DOI: https://doi.org/10.1109/TPAMI.2016.2577031.

ROBBINS, H.; MONRO, S. A Stochastic Approximation Method. **Annals of Mathematical Statistics**, v. 22, n. 3, p. 400–407, 1951.

SAPKOTA, R.; FLORES CALERO, M.; QURESHI, R.; BADJUGAR, C.; NEPAL, U.; POULOSE, A.; ZENO, P.; VADDEVOLU, U. B. P.; KHAN, S.; SHOMAN, M.; YAN, H.; KARKEE, M. YOLO advances to its genesis: a decadal and comprehensive review of the You Only Look Once (YOLO) series. **Artificial Intelligence Review**, v. 58, n. 274, 2025. DOI: https://doi.org/10.1007/s10462-025-11253-3.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. **Journal of Big Data**, v. 6, n. 1, p. 60, 2019. DOI: https://doi.org/10.1186/s40537-019-0197-0.

SOLOMON, C.; BRECKON, T. **Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab**. Chichester, UK: Wiley, 2011.

TAN, M.; PANG, R.; LE, Q. V. EfficientDet: Scalable and Efficient Object Detection. *In:* PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, USA: IEEE, 2020. p. 10778–10787. DOI: https://doi.org/10.1109/CVPR42600.2020.01079.

THEN, T. **Head Detection (CCTV) Dataset**. Des Moines, IA, USA, 2023. Disponível em: https://universe.roboflow.com/trisha-then/head-detection-cctv.

TIAN, Y.; YE, Q.; DOERMANN, D. YOLOv12: Attention-Centric Real-Time Object Detectors. **arXiv preprint arXiv:2502.11030**, 2025. Disponível em: https://arxiv.org/abs/2502.11030.

TIELEMAN, T.; HINTON, G. **Lecture 6.5 - RMSProp: Divide the gradient by a running average of its recent magnitude**. Mountain View, CA, USA, 2012. Disponível em: https://www.coursera.org/learn/neural-networks.

TORRALBA, A.; EFROS, A. A. Unbiased look at dataset bias. *In:* PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Providence, RI: [*s. n.*], 2011. p. 1521–1528. DOI: https://doi.org/10.1109/CVPR.2011.5995347.

TZUTALIN. **LabelImg: Label image bounding boxes for object detection**. Taipei, Taiwan, 2015. Disponível em: https://github.com/tzutalin/labelImg.

ULTRALYTICS. **Models Supported by Ultralytics**. Washington, USA, 2025. Disponível em: https://docs.ultralytics.com/pt/models/yolo12/.

VIDEEZY. **Seaport Worker Walking Towards the Camera**. California, USA, 2025. Disponível em: https://www.videezy.com/people/46484-seaport-worker-walking-towards-the-camera.

VUKICEVIC, A. M.; DJAPAN, M.; ISAILOVIC, V.; MILASINOVIC, D.; SAVKOVIC, M.; MILOSEVIC, P. Generic compliance of industrial PPE by using deep learning techniques. **Safety Science**, v. 148, p. 105646, 2022. DOI: https://doi.org/10.1016/j.ssci.2021.105646.

WANG, Z.; WU, Y.; YANG, L.; THIRUNAVUKARASU, A.; EVISON, C.; ZHAO, Y. Fast Personal Protective Equipment Detection for Real Construction Sites Using Deep Learning Approaches. **Sensors**, v. 21, p. 3478, 2021. DOI: https://doi.org/10.3390/s21103478.

WOJKE, N.; BEWLEY, A.; PAULUS, D. Simple Online and Realtime Tracking with a Deep Association Metric. *In:* PROCEEDINGS of the 2017 IEEE International

Conference on Image Processing (ICIP). Beijing, China: IEEE, 2017. p. 3645–3649. DOI: https://doi.org/10.1109/ICIP.2017.8296962.

WU, J.; CAI, N.; CHEN, W.; WANG, H.; WANG, G. Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset. **Automation in Construction**, v. 106, p. 102894, 2019. DOI: https://doi.org/10.1016/j.autcon.2019.102894.

XU, Z.; HUANG, J.; HUANG, K. A novel computer vision-based approach for monitoring safety harness use in construction. **IET Image Processing**, 2022. DOI: https://doi.org/10.1049/ipr2.12696.

YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; LIPSON, H. How Transferable Are Features in Deep Neural Networks? *In:* PROCEEDINGS of the 27th International Conference on Neural Information Processing Systems (NeurIPS). Montreal, QC, Canada: Curran Associates, Inc., 2014. p. 3320–3328.

YU, F.; LI, J.; WANG, X.; WU, S.; ZHANG, J.; ZENG, Z. Large, Complex, and Realistic Safety Clothing and Helmet Detection: Dataset and Method. **arXiv preprint arXiv:2306.02098**, 2024. Disponível em: https://arxiv.org/abs/2306.02098.

ZAIDI, S. F. A.; YANG, J.; ABBAS, M. S.; HUSSAIN, R.; LEE, D.; PARK, C. Vision-Based Construction Safety Monitoring Utilizing Temporal Analysis to Reduce False Alarms. **Buildings**, v. 14, p. 1878, 2024. DOI: https://doi.org/10.3390/buildings14061878.

ZHANG, H.; YAN, X.; LI, H.; JIN, R.; FU, H. Real-Time Alarming, Monitoring, and Locating for Non-Hard-Hat Use in Construction. **Journal of Construction Engineering and Management**, v. 145, n. 3, 2019. DOI: https://doi.org/10.1061/(ASCE)CO.1943-7862.0001629.

ZOU, Z.; CHEN, K.; SHI, Z.; GUO, Y.; YE, J. Object Detection in 20 Years: A Survey. **Proceedings of the IEEE**, v. 111, n. 3, p. 257–276, 2023. DOI: https://doi.org/10.1109/JPROC.2023.3238524.

APÊNDICES

APÊNDICE A - CÓDIGO DE AVALIAÇÃO DOS MODELOS

Abaixo está o código utilizado para avaliar o modelo YOLO com base nos datasets referenciados:

```
from ultralytics import YOLO
model = YOLO("/content/drive/MyDrive/Colab Notebooks/SFCHD_YOLOv11n/
    checkpoints/yolov11n/weights/best.pt")
results = model.val(data='/content/drive/MyDrive/Colab Notebooks/
    SFCHD_YOLOv11n/data_test.yaml')
metrics = results.results_dict
print(f"Precision: {metrics['metrics/precision(B)']:.4f}")
print(f"Recall: {metrics['metrics/recall(B)']:.4f}")
print(f"mAP50: {metrics['metrics/mAP50(B)']:.4f}")
```

APÊNDICE B - CÓDIGO DE AVALIAÇÃO DA FILTRAGEM COM RASTREIO

Abaixo está o código utilizado para avaliar o código de filtragem por quadros sequênciais no tempo. Para os demais modos de filtragem é alterado o tipo de rastreador ou ele é removido.

```
#@title Teste com ReID BotSORT
import cv2
import torch
import numpy as np
import time
from pathlib import Path
from ultralytics import YOLO
from boxmot import BotSort

# Configuracoes
model_path = "/content/best.pt"
video_path = "/content/teste3.mp4"
#video_path = "/content/teste4_7.5fps.mp4"
frame_rate = 30
alarm_threshold = 21
history_size = 30
quadros_com_erro = [(425, 440), (700, 715)]
batch_size = 8

device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
model = YOLO(model_path).to(device).eval()

tracker = BotSort(
    reid_weights=Path("osnet_x0_25_msmt17.pt"),
    device=device,
    half=True,
    per_class=False,
    track_high_thresh=0.5,
    track_low_thresh=0.5,
    new_track_thresh=0.5,
    track_buffer=5,
    match_thresh=0.4,
    proximity_thresh=0.5,
    appearance_thresh=0.5,
    cmc_method="ecc",
    frame_rate=frame_rate,
    fuse_first_associate=False,
    with_reid=True
```

```
)

vid = cv2.VideoCapture(video_path)
if not vid.isOpened():
    raise FileNotFoundError(f"Nao foi possivel abrir o video: {
        video_path}")

frame_counter = 0 # para calculo do FPS
id_class_count = {}   # para contagem das quantias dentro da janela
    deslizante
batch_frames = []

start_time = time.time()
try:
    while True:
        ret, frame = vid.read()
        if not ret:
            break

        batch_frames.append(frame)  #para acumulo antes de aplicar a GPU

        # So processa o batch quando atingir o tamanho
        if len(batch_frames) == batch_size:
            results = model.predict(batch_frames, conf=0.5, verbose=
                False, device=device)

            for i, result in enumerate(results):
                frame_counter += 1
                frame_i = batch_frames[i]

                # Simular erro forcando classe 0 dentro do intervalo
                    estipulado
                if any(start <= frame_counter <= end for (start, end) in
                    quadros_com_erro):
                    if len(result.boxes) > 0:
                        data_copy = result.boxes.data.clone()
                        data_copy[:, 5] = 0
                        result.boxes.data = data_copy

                # Conversao para [x1, y1, x2, y2, conf, cls]
                dets = []
                for box in result.boxes:
                    xyxy = box.xyxy.cpu().numpy()[0]
                    conf = float(box.conf.cpu())
                    cls = int(box.cls.cpu())
                    dets.append([*xyxy, conf, cls])
                # passagem para a classe do rastreador
```

```
                outputs = tracker.update(np.array(dets), frame_i)

                for obj in outputs:
                    track_id = int(obj[4])
                    class_id = int(obj[6])

                    # Historico de classes por ID
                    id_class_count.setdefault(track_id, []).append(
                        class_id)
                    id_class_count[track_id] = id_class_count[track_id
                        ][-history_size:]

                    # Alarme para classe 0
                    if id_class_count[track_id].count(0) >=
                        alarm_threshold:
                        timestamp = frame_counter / frame_rate
                        minutos = int(timestamp // 60)
                        segundos = timestamp % 60
                        print(f"ALERTA: ID {track_id} com {
                            alarm_threshold} ocorrencias da classe 0!
                            Tempo: {minutos:02d}:{segundos:05.2f} s")
                        id_class_count[track_id] = []  # evita alarmes
                            duplicados

            batch_frames = []

    # Processar frames restantes no final
    if batch_frames:
        results = model.predict(batch_frames, conf=0.5, verbose=False,
            device=device)
        for i, result in enumerate(results):
            frame_counter += 1
            frame_i = batch_frames[i]

            if any(start <= frame_counter <= end for (start, end) in
                quadros_com_erro):
                if len(result.boxes) > 0:
                    data_copy = result.boxes.data.clone()
                    data_copy[:, 5] = 0
                    result.boxes.data = data_copy

            dets = []
            for box in result.boxes:
                xyxy = box.xyxy.cpu().numpy()[0]
                conf = float(box.conf.cpu())
                cls = int(box.cls.cpu())
                dets.append([*xyxy, conf, cls])
```

```
            outputs = tracker.update(np.array(dets), frame_i)

            for obj in outputs:
                track_id = int(obj[4])
                class_id = int(obj[6])
                id_class_count.setdefault(track_id, []).append(class_id)
                id_class_count[track_id] = id_class_count[track_id][-
                    history_size:]

                if id_class_count[track_id].count(0) >= alarm_threshold:
                    timestamp = frame_counter / frame_rate
                    minutos = int(timestamp // 60)
                    segundos = timestamp % 60
                    print(f"ALERTA: ID {track_id} com {alarm_threshold}
                        ocorrencias da classe 0! Tempo: {minutos:02d}:{
                        segundos:05.2f} s")
                    id_class_count[track_id] = []

finally:
    total_time = time.time() - start_time
    avg_fps = frame_counter / total_time if total_time > 0 else 0

    print("\nProcessamento concluido!")
    print(f"Total de quadros: {frame_counter}")
    print(f"Tempo total: {total_time:.2f} segundos")
    print(f"FPS do video: {frame_rate}")
    print(f"FPS de processamento: {avg_fps:.2f}")

    vid.release()
```

APÊNDICE C - CÓDIGO DE TESTES DE TEMPO DE PROCESSAMENTO

Abaixo está o código utilizado para avaliar o tempo de processamento das filtragens. Neste exemplo está configurado o StrongSORT.

```
#@title StrongSort     V deo 8
import cv2
import torch
import numpy as np
import time
from pathlib import Path
from ultralytics import YOLO
from boxmot import StrongSort
import gc

model_path = "/content/HelmetDetection/models/best.pt"
video_path = "/content/video8.mp4"

batch_size = 8
frame_rate = 30
alarm_threshold = 21
history_size = 30

device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

model = YOLO(model_path).to(device).eval()

# Warm-up GPU
dummy_input = [np.zeros((640, 640, 3), dtype=np.uint8) for _ in range(
    batch_size)]
model.predict(dummy_input, conf=0.5, verbose=False, device=device)

tracker = StrongSort(
    reid_weights=Path('osnet_x0_25_msmt17.pt'),
    device=device,
    half=True,
    max_cos_dist=0.3,
    max_iou_dist=0.7,
    max_age=4,
    n_init=3,
    nn_budget=5,
    mc_lambda=0.95,
    ema_alpha=0.4,
    per_class=False
```

```
)

vid = cv2.VideoCapture(video_path)
if not vid.isOpened():
    raise RuntimeError(f"N o foi poss vel abrir o v deo: {video_path}
        ")

frame_counter = 0
id_class_count = {}
batch_frames = []

start_time = time.time()

while True:
    ret, frame = vid.read()
    if not ret:
        break

    batch_frames.append(frame)

    if len(batch_frames) == batch_size:
        results = model.predict(batch_frames, conf=0.5, verbose=False,
            device=device)

        for i, result in enumerate(results):
            frame_counter += 1
            frame_i = batch_frames[i]

            dets = []
            for box in result.boxes:
                xyxy = box.xyxy.cpu().numpy()[0]
                conf = float(box.conf.cpu())
                cls = int(box.cls.cpu())
                dets.append([*xyxy, conf, cls])

            outputs = tracker.update(np.array(dets), frame_i)

            for obj in outputs:
                track_id = int(obj[4])
                class_id = int(obj[6])

                id_class_count.setdefault(track_id, []).append(class_id)
                id_class_count[track_id] = id_class_count[track_id][-
                    history_size:]

                if id_class_count[track_id].count(0) >= alarm_threshold:
                    timestamp = frame_counter / frame_rate
```

```
                    minutos = int(timestamp // 60)
                    segundos = timestamp % 60
                    print(
                        f"ALERTA: ID {track_id} com {alarm_threshold}
                            ocorr ncias da classe 0 "
                        f"Tempo: {minutos:02d}:{segundos:05.2f} s"
                    )
                    id_class_count[track_id] = []

        batch_frames = []

if batch_frames:
    results = model.predict(batch_frames, conf=0.5, verbose=False,
        device=device)

    for i, result in enumerate(results):
        frame_counter += 1
        frame_i = batch_frames[i]

        dets = []
        for box in result.boxes:
            xyxy = box.xyxy.cpu().numpy()[0]
            conf = float(box.conf.cpu())
            cls = int(box.cls.cpu())
            dets.append([*xyxy, conf, cls])

        outputs = tracker.update(np.array(dets), frame_i)

        for obj in outputs:
            track_id = int(obj[4])
            class_id = int(obj[6])

            id_class_count.setdefault(track_id, []).append(class_id)
            id_class_count[track_id] = id_class_count[track_id][-
                history_size:]

            if id_class_count[track_id].count(0) >= alarm_threshold:
                timestamp = frame_counter / frame_rate
                minutos = int(timestamp // 60)
                segundos = timestamp % 60
                print(
                    f"ALERTA: ID {track_id} com {alarm_threshold}
                        ocorr ncias da classe 0 "
                    f"Tempo: {minutos:02d}:{segundos:05.2f} s"
                )
                id_class_count[track_id] = []
```

```
total_time = time.time() - start_time
avg_fps = frame_counter / total_time if total_time >0 else 0

print(f"Total de quadros: {frame_counter}")
print(f"Tempo total: {total_time:.2f} s")
print(f"FPS m dio de processamento: {avg_fps:.2f}")

vid.release()
del vid, batch_frames, id_class_count, tracker, model
gc.collect()
torch.cuda.empty_cache()
```

ANEXOS

ANEXO A - ACESSO AO DATASET

O dataset utilizado neste projeto pode ser acessado em: `https://www.kaggle.com/datasets/italostressersantos/shdataset`

ANEXO B - ACESSO AO DIRETÓRIO DO PROJETO NO GITHUB

O código-fonte e demais arquivos do projeto estão disponíveis em: `https://github.com/ItaloSSantos/HelmetDetection`

ANEXO C - ACESSO AO DIRETÓRIO DOS ARQUIVOS NO GOOGLE DRIVE

Os dados de treinamento, os modelos treinados nos 3 conjuntos, detecções realizadas e gráficos gerados podem ser acessados pelo endereço disponível:

https://drive.google.com/drive/folders/1_61cK2ULfYUNAdkA8km5h59dAp_Bu4EU?usp=sharing

ANEXO D - ARTIGO APRESENTADO NO XVII SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE

Automação de Inspeções de EPIs com Inteligência Artificial: estudos e desenvolvimentos para cenários reais

Ítalo S. Santos* Maria B. M. França**

* *Programa de Pós Graduação em Engenharia Elétrica, Universidade Estadual de Londrina, PR, (e-mail: italo.stresser@uel.br).*
** *Departamento de Engenharia Elétrica, Universidade Estadual de Londrina, PR (e-mail:mbmorais@uel.br)*

Abstract: This paper presents the development of a computer vision system for the automated inspection of safety helmet compliance, based on the YOLOv11n architecture. The system employs temporal filtering and object tracking techniques (BoT-SORT and StrongSORT) to reduce false alarms, while the custom dataset created for training aims to improve model accuracy. Preliminary results show that the model trained with the new dataset outperformed others in terms of precision and recall. However, challenges such as false positives/negatives in complex scenarios (e.g., hats or distance from the camera) still persist. The trackers proved effective in reducing false alarms but increased processing time. Future work includes the implementation of new filters and optimizations for real-time scenarios, focusing on the use of multiple monitoring channels.

Resumo: Este trabalho propõe um sistema de visão computacional para inspeção automatizada do uso de capacetes de segurança, utilizando a arquitetura YOLOv11n. O sistema emprega técnicas de filtragem temporal e rastreamento de objetos (BoT-SORT e StrongSORT) para reduzir falsos alarmes, enquanto o *dataset* criado especificamente para treinamento visa aumentar a precisão do modelo. Resultados preliminares mostram que o modelo treinado com o novo *dataset* superou outros em métricas de precisão e revocação. No entanto, desafios como falsos positivos/negativos em cenários complexos (e.g., uso de bonés ou distanciamento da câmera) ainda persistem. Os rastreadores demonstraram eficácia na redução de alarmes indevidos, mas impactaram o tempo de processamento. Trabalhos futuros incluem a implementação de novos filtros e otimizações para cenários de tempo real, com foco no uso de múltiplos canais de monitoramento.

Keywords: PPE Detection; Safety Helmet; Object Tracking; Prevention Systems.
Palavras-chaves: Detecção de EPI; Capacete de Segurança; Rastreamento de Objetos; Sistemas de Prevenção.

1. INTRODUÇÃO

A Organização Internacional do Trabalho estima que 2,78 milhões de pessoas morrem anualmente devido a acidentes ou doenças ocupacionais, e 374 milhões sofrem acidentes não fatais (ILO, 2021). No Brasil, os casos aumentaram de 465.770 em 2020 para 648.366 em 2022, conforme o Anuário Estatístico de Acidentes do Trabalho (MPS, 2022).

Nesse contexto, o uso de Equipamentos de Proteção Individual (EPIs) tem alta relevância, mas é frequentemente negligenciado. Baldissone et al. (2019) destacam que a falta de fiscalização reforça comportamentos inseguros, aumentando os riscos para os trabalhadores. A fiscalização presencial exige deslocamento contínuo e pode não cobrir todas as áreas de risco, enquanto o videomonitoramento, embora amplie a cobertura, depende da observação humana e é suscetível a falhas ou inviável em termos de recursos humanos.

Diante disso, este artigo apresenta o desenvolvimento de um sistema de visão computacional como tecnologia assistiva para a segurança no trabalho, com foco na verificação automática e em tempo real do uso de capacetes de segurança. O objetivo é viabilizar automações preventivas e ações de conscientização, contribuindo para a conformidade no uso de EPIs. O sistema inclui técnicas para reduzir falsos alarmes e é projetado para ser eficiente em termos de custo computacional, tornando-o adequado para aplicação prática em cenários com câmeras de vigilância.

Este documento está organizado da seguinte forma: a Seção 2 descreve uma revisão bibliográfica sobre avanços da visão computacional aplicada à supervisão automática do uso de EPIs; a Seção 3 detalha a metodologia experimental, incluindo configurações, softwares e dados utilizados; a Seção 4 apresenta os resultados obtidos nos

* O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

experimentos; e a Seção 5 expõe as conclusões e propõe novos experimentos e análises.

2. TRABALHOS RELACIONADOS

Com o avanço das redes neurais profundas e convolucionais (LeCun et al., 2015), aliado à evolução dos recursos de *hardware* voltados ao processamento paralelo, abordagens baseadas nessas técnicas têm sido exploradas na área de visão computacional, incluindo aplicações voltadas à segurança no trabalho.

2.1 Métodos com visão computacional

Na inspeção de EPIs, o detector Faster R-CNN (Ren et al., 2017) atinge bons desempenhos de precisão, enquanto modelos como o SSD (Liu et al., 2016) e o YOLO (Redmon et al., 2016) em suas versões recentes (Ali and Zhang, 2024), são mais indicados para aplicações em tempo real. Estudos comparativos e aprimoramentos desses algoritmos têm impulsionado o desenvolvimento de soluções mais práticas. Para fins de organização, a Tabela 1 apresenta os principais eixos de pesquisa em detecção de EPIs por visão computacional, conforme descrito a seguir.

Arquiteturas (1): Pesquisas buscam aprimorar arquiteturas de detectores, otimizando algoritmos para melhor desempenho em métricas de validação, como precisão e velocidade de inferência.

Dataset (2): Estudos focam na criação e aprimoramento de conjuntos de imagens para treinamento, incluindo rotulagem, aumento de dados e técnicas que melhoram o aprendizado.

Método de inferência (3): Abordagens de inferência são analisadas, como a detecção direta de EPIs em trabalhadores ou a verificação de sua associação ao corpo, além da avaliação de riscos relacionados à falta de EPIs durante o trabalho.

Tempo real (4): Pesquisas exploram implementações para hardware limitado, como dispositivos de borda, e a capacidade dos modelos de operar em tempo real.

Prevenção (5): Com a viabilidade da operação em tempo real, estudos avançam no desenvolvimento de automações para prevenção e conscientização sobre o uso de EPIs.

Como evidenciado na Tabela 1, os desenvolvimentos na área seguem diferentes vertentes. Alguns estudos apresentam similaridades com este trabalho e são descritos na próxima seção.

Tabela 1. Classificação das referências segundo as cinco categorias de estudos e desenvolvimentos.

Referência	1	2	3	4	5
Fang et al. (2018)	✓	✓			
Wu et al. (2019)	✓	✓			
Nath et al. (2020)		✓	✓	✓	
Wang et al. (2021)		✓			
Otgonbold et al. (2022)		✓			
Ke et al. (2022)	✓	✓		✓	
Vukicevic et al. (2022)			✓		
Xu et al. (2022)	✓		✓	✓	✓
Hu and Ren (2023)	✓				
Lee and Yu (2023)			✓		✓
An et al. (2023)	✓			✓	
Lo et al. (2023)		✓			
Lee et al. (2023)	✓		✓		
Lema et al. (2023)			✓	✓	✓
Ahmad and Rahimi (2024)		✓			
Di et al. (2024)	✓	✓			
Lin (2024)	✓			✓	
Zaidi et al. (2024)			✓		✓

2.2 Implementações para automação preventiva

Xu et al. (2022) aprimoraram o YOLOv5 para detectar cintos de segurança com maior velocidade e acurácia. Propuseram um sistema preventivo com múltiplas câmeras e minicomputadores interligados por rede sem fio, permitindo alertas em tempo real. No entanto, sua eficácia poderia ser ampliada com filtros para reduzir falsos positivos e validação em vídeos de cenários reais.

Lema et al. (2023) implementaram rastreadores de objetos para minimizar falsos alarmes em uma inspeção automatizada de EPIs, utilizando minicomputadores NVIDIA Jetson Nano. Embora mais acessível que outras GPUs, seu uso em larga escala pode tornar o projeto oneroso. Como alternativa, sugere-se um sistema de menor custo, compartilhando processamento entre múltiplos canais de câmeras, exigindo ajustes nos parâmetros de rastreamento.

Zaidi et al. (2024) reduziram erros de detecção por meio de uma análise temporal, combinando detecção de objetos com um módulo específico, resultando em menor taxa de falsos alarmes e maior confiabilidade. Esse método pode ser uma alternativa ao rastreamento, mas comparações adicionais são necessárias para avaliar sua eficácia.

3. MÉTODOS

O método adotado baseou-se em uma revisão bibliográfica, apresentada na Seção 2, que permitiu identificar e analisar as melhores estratégias para o problema. Com base nessa análise, destacam-se três frentes de investigação para aprimorar a inspeção automatizada de EPIs:

- Desenvolvimento e disponibilização de um *dataset* generalista e balanceado para a detecção da conformidade no uso de capacetes;
- Comparação do desempenho de rastreadores, como BoT-SORT (Aharon et al., 2022) e StrongSORT (Du et al., 2023), além da aplicação de filtragens temporais no sistema de inspeção;
- Análise do desempenho do sistema considerando a flexibilização do processamento em tempo real, por meio da variação da taxa de quadros por segundo e da filtragem agregada.

A partir dessas diretrizes, a construção do sistema de visão computacional utilizado nos testes seguiu algumas etapas, abrangendo a criação do *dataset* até a implementação e avaliação de rastreadores. Essas etapas estão ilustradas na Figura 1 e detalhadas nas Seções 3.1 a 3.3.

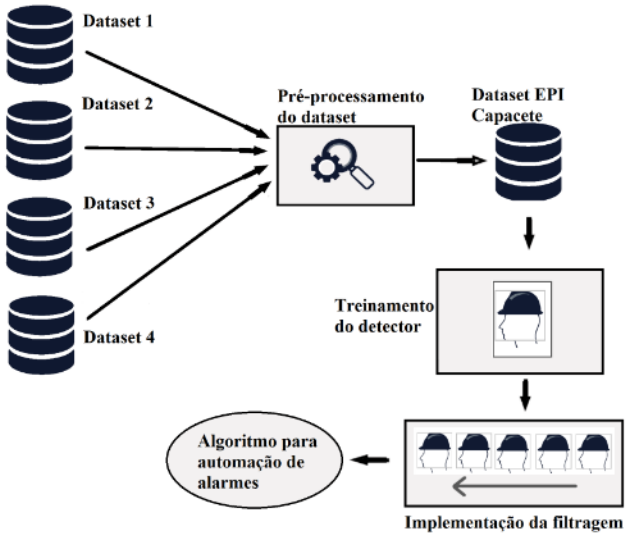


Figura 1. Procedimentos para a elaboração do sistema de visão computacional para inspeção automática de EPIs.

3.1 Pré-processamento e geração do dataset

O *dataset* gerado teve intuito de ser mais abrangente que os disponíveis publicamente e com imagens em cenários reais de câmeras de vigilância. Para isso, o *dataset* base utilizado foi o SFCHD (Yu et al., 2024), que contém imagens anotadas de pessoas com e sem capacete em cenários industriais. Ele foi carregado na plataforma Roboflow® (Dwyer et al., 2024). As classes foram organizadas como "com capacete" e "sem capacete", com detecção baseada na região completa da cabeça. Também foi utilizado o *dataset* de (Otgonbold et al., 2022), que segue a mesma abordagem, permitindo aumentar a quantidade de instâncias e melhorar a generalização das imagens.

No entanto, ambos os conjuntos apresentavam desbalanceamento de classes, o que poderia acarretar viés no treinamento do modelo. Para corrigir isso, foram adicionadas imagens da classe "sem capacete" dos *datasets* HeadDetectionCCTV (Then, 2023) e SCUT-HEAD (Peng et al., 2018), como também aplicadas técnicas de *augmentation* (Claro et al., 2020). Além da melhoria do balanceamento, o *augmentation* foi implementado para aumentar a variabilidade do conjunto e simular condições visuais distintas. Essas variações visam melhorar a capacidade de generalização do modelo, tornando-o mais robusto a situações de cenários reais. As técnicas aplicadas incluíram inversão horizontal, rotação (±15°), cisalhamento (±10°), ajuste de brilho (-25% a +25%), desfoque (até 2px) e adição de ruído (até 0,7% dos pixels).

Por fim, obteve-se uma quantidade de instâncias mais balanceadas e com imagens com diversas características e fundos. A Figura 2 mostra a quantidade de instâncias das duas classes para o *dataset* criado e nomeado por SH (safety helmet), cujo link está disponível no Apêndice A.

3.2 Escolha do detector e treinamento

Estudos como Hu and Ren (2023); Lee et al. (2023); Lin (2024) embasaram a escolha da arquitetura YOLO por equilibrar acurácia e velocidade. Optou-se pela versão YOLOv11n (Jocher and Qiu, 2024), que oferece inferência rápida com bom desempenho de detecção.

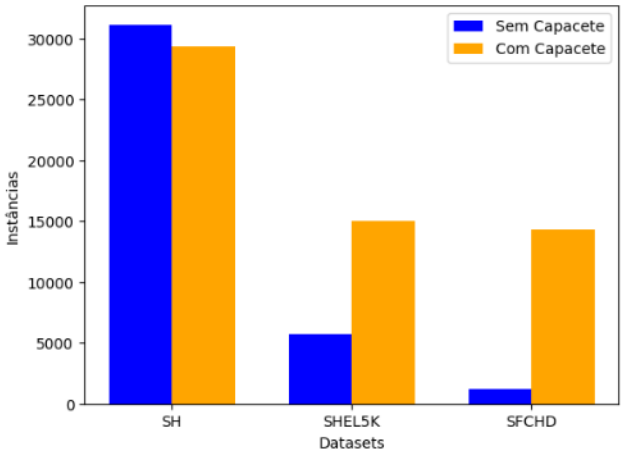


Figura 2. Distribuição das classes com ou sem capacete, quantificadas por instâncias em cada *dataset* .

Para desenvolver o modelo de detecção, a plataforma Colab (*Colaboratory*) do Google® foi utilizada. Esse ambiente é baseado em Jupyter Notebook com acesso a GPUs, o que acelera o treinamento e testes dos modelos.

Durante o treinamento e os testes, utilizaram-se as bibliotecas da Ultralytics® (Ultralytics, 2025), que auxiliam a implementação e o ajuste do modelo de detecção. A partir delas, foi importado o YOLO pré-treinado, utilizando a técnica *Transfer Leaning*(Ribani and Marengoni, 2019). Para os testes iniciais, optou-se pelo otimizador SGD, baseado em comparações com o AdamW e nos dados apresentados por Zaidi et al. (2024). Para evitar o sobreajuste (*overfitting*) e desperdício de computacional, foi configurado o parâmetro *patience* que refere-se ao conceito de parada antecipada (*early stoping*) (Prechelt, 1998).

O treinamento foi realizado em uma GPU NVIDIA® T4 (15GB), o que permitiu variar o *batch size* nos testes. A Tabela 2 apresenta os hiperparâmetros do modelo final.

Tabela 2. Hiperparâmetros utilizados no treinamento do modelo de detecção.

Parâmetro	Valor
epochs	80
batch	64
imgsz	640
optimizer	SGD
patience	10

Para avaliar o desempenho do modelo, utilizou-se a métrica *mAP@0.5*, que corresponde à média da *Average Precision* (AP) de cada classe (uso e não uso de capacete). A AP é calculada como a área sob a curva *Precision-Recall*, construída a partir das detecções consideradas corretas quando a interseção com a anotação de referência (*IoU*) é igual ou superior a 0,5. Essa métrica é amplamente utilizada na avaliação de detectores de objetos (Everingham et al., 2010). Os valores de *mAP@0.5*, *Precision* e *Recall* obtidos durante o treinamento podem ser visualizados na Figura 3.

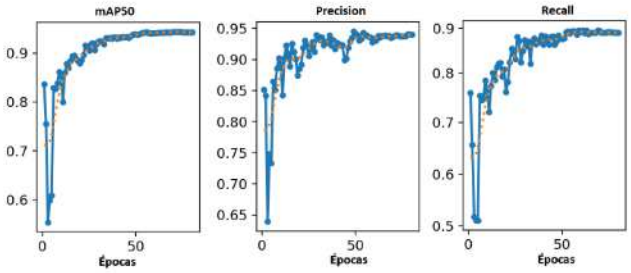


Figura 3. Evolução das métricas de desempenho (*mAP@0.5*, *Precision* e *Recall*) durante o treinamento do modelo.

3.3 Algoritmo para Filtragem

O algoritmo de filtragem de detecções, baseado em Zaidi et al. (2024), utiliza um *buffer* dinâmico que armazena os resultados recentes. A cada novo quadro, calcula-se a porcentagem de detecções "sem capacete" no *buffer*, e um alarme é acionado apenas se esse valor ultrapassar um limiar (ex.: 70%), reduzindo falsos positivos por detecções instáveis.

Outra forma de filtragem utilizada foi pelos rastreadores de múltiplos objetos, de forma semelhante ao trabalho de Lema et al. (2023). Através da atribuição de identificadores únicos (IDs) e preservando a classe de cada objeto os algoritmos de BoT-SORT e StrongSORT Broström (2023), possibilitam a filtragem individualizada dos objetos detectados (pessoas utilizando EPI ou não), assim gerando um acompanhamento contínuo através do quadros de imagem, eliminando oscilações de inferências errôneas, ou também eliminando falsos alarmes causados por objetos estáticos.

Essas técnicas foram testadas em vídeos realizados no campus e retirados da internet, avaliando erros que gerariam alarmes indevidos, a eficácia das filtragens e o tempo médio de processamento por quadro.

4. RESULTADOS PRELIMINARES

Nesta seção, são apresentados os primeiros resultados obtidos com o treinamento e testes do modelo YOLO e com implementação dos algoritmos de filtragem.

4.1 Desempenho do modelo e dataset

Através do detector treinado foi possível avaliar seu desempenho e verificar se o *dataset* SH apresentava uma boa performance em diferentes imagens de teste. Como critério de comparação, também foram realizados treinamentos com os conjuntos referenciados na Seção 3, cujos resultados estão apresentados na Tabela 3.

O modelo com treino pelo SH apresentou ótima capacidade de detecção em testes tanto no SH quanto no SHEL5K. Apesar de um desempenho inferior no SFCHD, o resultado alcançado ainda foi relevante, com uma *mAP* superior em comparação ao treinamento realizado diretamente com o próprio *dataset* SFCHD.

Para avaliar o desempenho do modelo em condições reais, foram utilizados os vídeos coletados para inferência. Os quadros extraídos foram analisados quanto a falhas de

Tabela 3. Cruzamento de treino e teste do YOLO11n em diferentes *datasets* de uso de capacete.

Conjunto de treino	Medidas no conjunto de teste (P; R; mAP50)*		
	SH	SFCHD	SHEL5K
SH	0,94;0,89;0,94	0,85;0,81;0,86	0,93;0,90;0,95
SFCHD	0,85;0,62;0,70	0,78;0,68;0,77	0,61;0,50;0,53
SHEL5K	0,80;0,41;0,46	0,79;0,56;0,66	0,91;0,85;0,91

*P: Precisão ; R: Revocação; mAP@0.5: Média da Precisão Média com *IoU* de 50%.

detecção e rastreamento. Na Figura 4, observam-se casos de falsos positivos para a classe "com capacete" (4b e 4c). Isso se deve principalmente por cenários diversos formados ao longo dos quadros de imagem, que podem se assemelhar à classe inferida, ou também, pelo distanciamento e diminuição de tamanho do objeto a ser inferido.

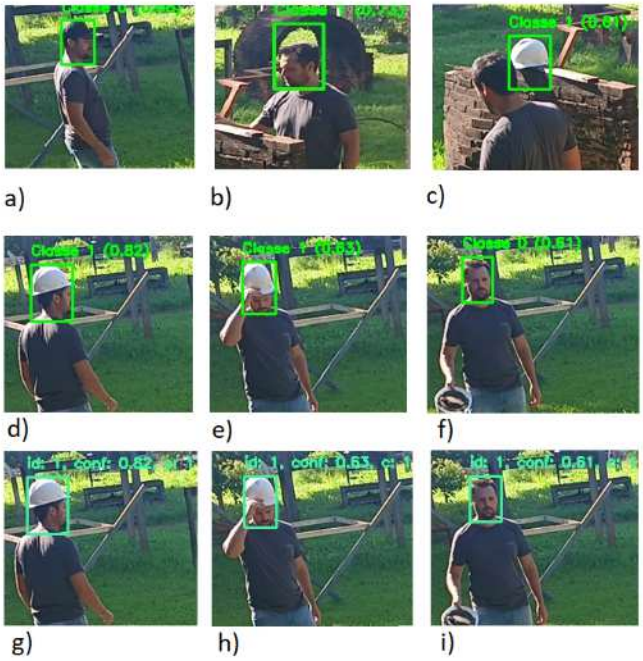


Figura 4. Exemplos de quadros processados pelo detector sem rastreador: a) até f). Os quadros g) até i) são com o rastreador BotSORT.

Além disso, no Vídeo 5 da Tabela 4, observou-se um número relevante de falsos negativos e positivos para ambas as classes. Esses casos ocorreram principalmente pelo indivíduo utilizar um boné (Figura 4a) durante a gravação, causando uma similaridade visual entre bonés e capacetes, o que resultou em quadros de difícil classificação.

Contudo, os erros que poderiam gerar falsos alarmes foram nulos nas gravações de melhor qualidade e com formato 1280x720p, presentes nos Vídeo2 a Vídeo5. Já para o Vídeo1, retirado da internet, com resolução formatada para 640x640, foram gerados seis falsos positivos.

4.2 Testes de filtragem

Para testes de melhoria do sistema, foi utilizado o algoritmo de filtragem por percentual em quadros sequenciais, denominado Filtro 1. Para sua aplicação, foi configurada uma fila de 20 classes detectadas e um percentual de 75%

Tabela 4. Quantidade de erros de detecção em quadros de vídeos processados com o detector sem filtros.

	Quadros	Sem Capacete		Com Capacete	
		(F.P.)	(F.N)	(F.P.)	(F.N)
Vídeo1	96	6	4	1	3
Vídeo2	902	0	n.a.*	0	85
Vídeo3	902	0	62	142	7
Vídeo4	902	0	1	6	45
Vídeo5	902	0	236	157	n.a.*

*n.a.: não aplicável devido a inexistência da classe.

para o acionamento do alarme. Esse filtro foi aplicado ao Vídeo 1 e ao Vídeo 4, e, com a utilização da GPU T4 conectada pelo Colab, os vídeos foram processados. Os resultados referentes ao tempo de processamento e à quantidade de alarmes gerados estão apresentados na Tabela 5.

Tabela 5. Métricas de processamento dos vídeos com o uso dos algoritmos de filtragem. As colunas indicam o tempo de processamento em segundos (T), a quantidade de alarmes gerados (A) e a taxa de quadros por segundo (QPS).

	Vídeo 1			Vídeo 4		
	T(s)	A	QPS	T(s)	A	QPS
Filtro 1	1.52	4	63	11.7	2	77
BotSORT	3.50	3	27,6	27.4	2	32
StrongSort	4.17	3	22,3	35.4	2	25,3

Nos vídeos analisados, conforme apresentado na Tabela 5, foram observados momentos em que a ausência do capacete foi detectada, gerando alarmes. Sobre esse mesmo ponto, os rastreadores introduziram histerese nas mudanças de classe e ID, Figuras 4g a 4i, reduzindo oscilações em cenários de baixa confiança. Isso pode ser quantificado pela pequena variação de alarmes no Vídeo 1 da Tabela 5.

Os rastreadores atuam como filtros para erros em situações onde a confiança é baixa e podem ocorrer oscilações de classes inferidas. Além disso, em cenários com múltiplas pessoas, eles permitem uma filtragem específica por objeto (pessoa no vídeo), melhorando a precisão do sistema. No entanto, apesar das melhorias, a utilização dos rastreadores resultou em um aumento no tempo de processamento, conforme evidenciado pelos testes iniciais apresentados na Tabela 5. Ambos os rastreadores, BotSORT e StrongSORT, causaram uma redução significativa na taxa de quadros por segundo, com diminuições de 58% e 66%, respectivamente, em comparação à taxa de quadros por segundo sem rastreadores.

5. CONCLUSÕES

Este trabalho propôs um sistema de visão computacional baseado em YOLOv11n para inspeção automatizada do uso de capacetes de segurança, atuando como tecnologia assistiva para a segurança no trabalho. O sistema integrou técnicas de filtragem temporal e rastreamento de objetos (BotSORT e StrongSORT), reduzindo falsos alarmes e alcançando alta precisão e revocação com o treinamento no *dataset* SH desenvolvido. Embora desafios como falsos positivos/negativos em cenários complexos e aumento no tempo de processamento (com reduções de 42% e 33% na taxa de quadros) tenham sido observados, o sistema mostrou-se eficaz ao evitar falsos alarmes, mesmo diante de oscilações causadas por falsas inferências de não uso de capacete.

Como trabalhos futuros, serão exploradas otimizações nos rastreadores, incluindo algoritmos sem reidentificação, filtragem baseada na média móvel da confiança das inferências e varredura de múltiplos canais, visando aprimorar o desempenho do sistema para os cenários reais.

AGRADECIMENTOS

Agradecemos à PROPPG-UEL e ao PPGEE-Mestrado pela oportunidade de realização deste trabalho. Também reconhecemos o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), Código de Financiamento 001.

REFERÊNCIAS

Aharon, N., Orfaig, R., and Bobrovsky, B.Z. (2022). Bot-sort: Robust associations multi-pedestrian tracking. URL https://arxiv.org/abs/2206.14651.

Ahmad, H.M. and Rahimi, A. (2024). Sh17: A dataset for human safety and personal protective equipment detection in manufacturing industry. *Journal of Safety Science and Resilience*. doi:10.1016/j.jnlssr.2024.09.002.

Ali, M.L. and Zhang, Z. (2024). The yolo framework: A comprehensive review of evolution, applications, and benchmarks in object detection. *Computers*, 13, 336. doi:10.3390/computers13120336.

An, Q., Xu, Y., Yu, J., Tang, M., Liu, T., and Xu, F. (2023). Research on safety helmet detection algorithm based on improved yolov5s. *Sensors*, 23(13), 5824. doi: 10.3390/s23135824.

Baldissone, G., Comberti, L., Bosca, S., and Murè, S. (2019). The analysis and management of unsafe acts and unsafe conditions: Data collection and analysis. *Safety Science*, 119, 240–251.

Broström, M. (2023). Boxmot: A collection of sota real-time, multi-object trackers for object detectors. doi: 10.5281/zenodo.8132989.

Claro, M., Vogado, L., Santos, J., and Veras, R. (2020). Utilização de técnicas de data augmentation em imagens: Teoria e prática. In *Minicursos da ERCEMAPI 2020*. doi:10.5753/sbc.11.5.3. URL https://books-sol.sbc.org.br/index.php/sbc/catalog/view/48/217/455. Acessado em: 17/02/2025.

Di, B., Xiang, L., Daoqing, Y., and Kaimin, P. (2024). Mara-yolo: An efficient method for multiclass personal protective equipment detection. *IEEE Access*, 12, 24866–24878. doi:10.1109/ACCESS.2024.3365504.

Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., and Meng, H. (2023). Strongsort: Make deepsort great again. URL https://arxiv.org/abs/2202.13514.

Dwyer, B., Nelson, J., Hansen, T., et al. (2024). Roboflow (version 1.0) [software]. https://roboflow.com/. Computer Vision.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2), 303–338.

Fang, Q., Li, H., Luo, X., Ding, L., Luo, H., Rose, T.M., and An, W. (2018). Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. *Automation in Construction*, 85, 1–9. doi:10.1016/j.autcon.2017.09.018.

Hu, L. and Ren, J. (2023). Yolo-lhd: An enhanced lightweight approach for helmet wearing detection in industrial environments. *Frontiers in Built Environment*, 9, 1288445. doi:10.3389/fbuil.2023.1288445.

ILO (2021). Nine business practices for improving safety and health through supply chains and building a culture of prevention and protection. URL www.ilo.org/sites/default/files/wcmsp5/groups/public/%40dgreports/%40dcomm/documents/briefingnote/wcms_821481.pdf. Accesso em: 17/02/2025.

Jocher, G. and Qiu, J. (2024). Ultralytics yolo11. URL https://github.com/ultralytics/ultralytics. Accesso em: 21/10/2024.

Ke, X., Chen, W., and Guo, W. (2022). 100+ fps detector of personal protective equipment for worker safety: A deep learning approach for green edge computing. *Peer-to-Peer Networking and Applications*, 15, 950–972. doi:10.1007/s12083-021-01258-4.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444. doi:10.1038/nature14539.

Lee, J.Y., Choi, W.S., and Choi, S.H. (2023). Verification and performance comparison of cnn-based algorithms for two-step helmet-wearing detection. *Expert Systems with Applications*, 225, 120096. doi:10.1016/j.eswa.2023.120096.

Lee, S.K. and Yu, J.H. (2023). Ontological inference process using ai-based object recognition for hazard awareness in construction sites. *Automation in Construction*, 153, 104961. doi:10.1016/j.autcon.2023.104961.

Lema, D., Usamentiaga, R., and García, D. (2023). Low-cost system for real-time verification of personal protective equipment in industrial facilities using edge computing devices. *Journal of Real-Time Image Processing*, 20, 111. doi:10.1007/s11554-023-01368-7.

Lin, B. (2024). Safety helmet detection based on improved yolov8. *IEEE Access*, 12, 28260–28272. doi:10.1109/ACCESS.2024.3368161.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., and Berg, A.C. (2016). Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, 9905, 21–37. doi:10.1007/978-3-319-46448-0_2.

Lo, J.H., Lin, L.K., and Hung, C.C. (2023). Real-time personal protective equipment compliance detection based on deep learning algorithm. *Sustainability*, 15, 391. doi:10.3390/su15010391.

MPS (2022). Anuário estatístico de acidentes de trabalho. URL https://www.gov.br/previdencia/pt-br/assuntos/previdencia-social/saude-e-seguranca-do-trabalhador/acidente_trabalho_incapacidade/arquivos/AEAT_2022/copy_of_subsecao-a-acidentes-do-trabalho/capitulo-59-brasil. Acesso em: fevereiro de 2025.

Nath, N.D., Behzadan, A.H., and Paal, S.G. (2020). Deep learning for site safety: Real-time detection of personal protective equipment. *Automation in Construction*, 112, 103085. doi:10.1016/j.autcon.2020.103085.

Otgonbold, M.E., Gochoo, M., Alnajjar, F., Ali, L., Tan, T.H., Hsieh, J.W., and Chen, P.Y. (2022). Shel5k: An extended dataset and benchmarking for safety helmet detection. *Sensors*, 22(6), 2315. doi:10.3390/s22062315.

Peng, D., Sun, Z., Chen, Z., Cai, Z., Xie, L., and Jin, L. (2018). Detecting heads using feature refine net and cascaded multi-scale architecture. *arXiv preprint arXiv:1803.09256*.

Prechelt, L. (1998). Early stopping - but when? *Neural Networks: Tricks of the Trade*, 53–67.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 779–788. doi:10.1109/CVPR.2016.91.

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137–1149. doi:10.1109/TPAMI.2016.2577031.

Ribani, R. and Marengoni, M. (2019). A survey of transfer learning for convolutional neural networks. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, 47–57. doi:10.1109/SIBGRAPI-T.2019.00010.

Then, T. (2023). Head detection (cctv) dataset. https://universe.roboflow.com/trisha-then/head-detection-cctv. URL https://universe.roboflow.com/trisha-then/head-detection-cctv. Visited on 2025-02-18.

Ultralytics (2025). Models supported by ultralytics. https://docs.ultralytics.com/models.

Vukicevic, A.M., Djapan, M., Isailovic, V., Milasinovic, D., Savkovic, M., and Milosevic, P. (2022). Generic compliance of industrial ppe by using deep learning techniques. *Safety Science*, 148, 105646. doi:10.1016/j.ssci.2021.105646.

Wang, Z., Wu, Y., Yang, L., Thirunavukarasu, A., Evison, C., and Zhao, Y. (2021). Fast personal protective equipment detection for real construction sites using deep learning approaches. *Sensors*, 21, 3478. doi:10.3390/s21103478.

Wu, J., Cai, N., Chen, W., Wang, H., and Wang, G. (2019). Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset. *Automation in Construction*, 106, 102894. doi:10.1016/j.autcon.2019.102894.

Xu, Z., Huang, J., and Huang, K. (2022). A novel computer vision-based approach for monitoring safety harness use in construction. *IET Image Processing*. doi:10.1049/ipr2.12696.

Yu, F., Li, J., Wang, X., Wu, S., Zhang, J., and Zeng, Z. (2024). Large, complex, and realistic safety clothing and helmet detection: Dataset and method. URL https://arxiv.org/abs/2306.02098.

Zaidi, S., Yang, J., Abbas, M., Hussain, R., Lee, D., and Park, C. (2024). Vision-based construction safety monitoring utilizing temporal analysis to reduce false alarms. *Buildings*, 14, 1878. doi:10.3390/buildings14061878.

Apêndice A. *DATASET* SH

https://www.kaggle.com/datasets/italostressersantos/shdataset