UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Bruno Cúgola Coelho

Aplicação de visão computacional em câmeras de segurança: Identificação de pessoas e objetos - detecção de níveis de perigo

Juiz de Fora
2025

Bruno Cúgola Coelho

Aplicação de visão computacional em câmeras de segurança: Identificação de pessoas e objetos - detecção de níveis de perigo

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Sistemas Eletrônicos

Orientador: Prof. André Luís Marques Marcato, D.Sc.

Coorientador: Prof. Iago Zanuti Biundini, D.Sc.

Juiz de Fora

2025

Bruno Cúgola Coelho

Aplicação de visão computacional em câmeras de segurança: Identificação de pessoas e objetos - detecção de níveis de perigo

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Sistemas Eletrônicos

Aprovada em 24 de julho de 2025

Bruno Cúgola Coelho

Aplicação de visão Computacional em Câmeras de Segurança: Identificação de Pessoas e Objetos - Detecção de Níveis de Perigo

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Sistemas Eletrônicos

Aprovada em 24 de julho de 2025.

BANCA EXAMINADORA

Prof. Dr. André Luis Marques Marcato - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Iago Zanuti Biundini - Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Vinícius Barbosa Schettino
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Alexandre Bessa dos Santos
Universidade Federal de Juiz de Fora

Juiz de Fora, 02/07/2025.

Documento assinado eletronicamente por **Vinicius Barbosa Schettino**, **Usuário Externo**, em 25/07/2025, às 11:23, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Andre Luis Marques Marcato**, **Professor(a)**, em 20/10/2025, às 07:55, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Iago Zanuti Biundini**, **Usuário Externo**, em 20/10/2025, às 09:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Alexandre Bessa dos Santos**, **Professor(a)**, em 20/10/2025, às 10:00, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **2478679** e o código CRC **C0BCE819**.

Dedico este trabalho à minha família, especialmente ao meu pai, Adilson, que faleceu no ano de 2024. Obrigado por sempre me apoiarem e por se fazerem presentes em todos os momentos.

Dedico também à Leatrice, que esteve comigo me auxiliando, corrigindo e dando apoio ao longo do caminho.

Também dedico ao meu orientador, Dr. André Marcato, e ao meu co-orientador, Iago Biundini, pela oportunidade, pelo companheirismo e pelas orientações oferecidas durante a execução da pesquisa.

Sem esses três pilares e muito estudo esse trabalho não teria sido finalizado.

AGRADECIMENTOS

Agradeço à Universidade Federal de Juiz de Fora e ao Programa de Pós-graduação em Engenharia Elétrica pela oportunidade de cursar o Mestrado e por todo o suporte oferecido.

Agradeço à FAPEMIG pelo fomento à esta pesquisa, que auxiliou na aquisição dos insumos para o trabalho, e ao CAPES pelo acesso a materiais de pesquisa e desenvolvimento.

RESUMO

A detecção de movimento de objetos por meio de câmeras de vigilância tem se mostrado uma ferramenta cada vez mais relevante em diversas áreas, como segurança pública, saúde, meio ambiente e monitoramento remoto. Essa tecnologia tem transformado a forma como ambientes são protegidos e como respostas rápidas a situações de risco são implementadas. O desenvolvimento de algoritmos autônomos aliados a técnicas de *Deep Learning*, como redes neurais convolucionais (CNNs) e o modelo *You Only Look Once* (YOLO), tem permitido a identificação precisa de armas, atos de violência e outros comportamentos suspeitos em tempo real. Esses sistemas possibilitam a redução da carga de trabalho dos operadores humanos e garantem maior eficiência nas operações de segurança e monitoramento. Entretanto, a aplicação de técnicas de detecção de objetos pode gerar incertezas e imprecisões, como falsos positivos (identificação de objetos inexistentes), falsos negativos (não detecção de objetos presentes), interferências visuais e variações ambientais.Esses fatores podem comprometer a confiabilidade das detecções e criar falhas no processo. Uma forma de lidar com essas incertezas e imprecisões é o ajuste do *treshold*, que foi ajustado até chegar ao valor de 0,47 permitindo manter um bom nível de precisão (89,85%), e a aplicação da lógica *fuzzy*, que, por meio da análise das funções de pertinência, avaliando velocidade e distância entre os objetos, permitiu uma melhora significativa na precisão final do projeto. Pois, foi possível interpretar e gerar alertas apenas quando necessário.

Palavras-chave: Sistemas de segurança eletrônica, câmeras de segurança, visão computacional, YOLOv8

ABSTRACT

The detection of object movement through surveillance cameras has proven to be an increasingly relevant tool in various fields, such as public security, healthcare, environmental monitoring, and remote surveillance. This technology has transformed the way environments are protected and how rapid responses to risk situations are implemented. The development of autonomous algorithms combined with Deep Learning techniques—such as Convolutional Neural Networks (CNNs) and the You Only Look Once (YOLO) model—has enabled the accurate identification of weapons, violent acts, and other suspicious behaviors in real time. These systems reduce the workload on human operators and ensure greater efficiency in security and monitoring operations. However, the application of object detection techniques may lead to uncertainties and inaccuracies, such as false positives (identifying non-existent objects), false negatives (failing to detect existing objects), visual interferences, and environmental variations. These factors can compromise the reliability of detections and lead to failures in the process. One way to address these uncertainties and inaccuracies is by adjusting the threshold, which was fine-tuned until reaching a value of 0.47, allowing the system to maintain a high level of precision (89.85%). Additionally, the application of fuzzy logic, through the analysis of membership functions—evaluating object speed and distance—enabled a significant improvement in the project's final accuracy, as the system was able to interpret and generate alerts only when necessary.

Keywords: Electronic security system; Surveillance cameras; Computer vision; YoloV8.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Networks*
YOLO	*You Only Look Once*
SSD	*Single Shot MultiBox Detector*
ROS	*Robot Operating System*
OpenCV	*Open Source Computer Vision Library*
mAP	*mean Average Precision*
NMS	*Non-Maximum Suppression*
IoT	*Internet of things* ou internet das coisas
FLOP	Floating Point Operations Per Second
IBM	International Business Machines Corporation
FPS	Quadros por segundo
PR	Precisão-Recall
IP	Protocolo de Internet

SUMÁRIO

1 Introdução

Com o crescimento dos centros urbanos e da densidade populacional, as taxas de criminalidade também se elevaram, enquanto os criminosos passaram a se beneficiar do anonimato proporcionado pela grande movimentação e pela falta de controle social. Nesse contexto, Sérgio Salomão Shecaira destaca que "a explosão de crescimento da cidade, que se expande em círculos do centro para a periferia, cria graves problemas sociais, trabalhistas, familiares, morais e culturais que se traduzem em um fermento conflituoso, potencializador da criminalidade" (Shecaira, 2004, *apud* Gomes, 2019, p.101). Esse cenário evidencia como o desenvolvimento urbano desordenado pode contribuir para a insegurança e dificultar a identificação de infratores [Gomes 2019], o que formou um ambiente marcado pelo medo e pela desconfiança entre a população [Lima, Misse e Miranda 2000]. As estatísticas criminais, por sua vez, refletem não apenas a violência em si, mas também revelam um quadro mais amplo, que abrange fatores sociais, econômicos e culturais, como a desigualdade social, o acesso precário à educação e a falta de oportunidades de trabalho [Fajnzylber e Junior 2001].

Além disso, em meados do século XIV, surgiram as armas de fogo portáteis. Inicialmente utilizadas como peças de artilharia de grande porte, muitas vezes exigiam montaria ou algum tipo de suporte para garantir estabilidade. Com o passar do tempo, essas armas se tornaram cada vez mais populares e foram sendo aperfeiçoadas, impulsionadas pela crescente necessidade de demonstração de força, tanto por parte dos Estados em formação quanto pela própria sociedade [Mardel 1887].

Nesse contexto, a identificação do uso de armas de fogo em situações de assaltos, roubos e homicídios começou a se tornar uma prioridade [Baldessar et al. 2021]. A presença de uma arma de fogo em uma situação de violência pode alterar rapidamente a dinâmica de um crime, tornando-o ainda mais perigoso e letal. Como resultado, novas tecnologias começaram a ser desenvolvidas para identificar e monitorar sua utilização em tempo real. Exemplos disso são os sistemas de segurança que surgiram como uma tentativa de reduzir as taxas de criminalidade e que rapidamente se espalharam pelo mundo entre o final do século XIX e início

do século XX, evoluindo para os sistemas automatizados utilizados atualmente. No entanto, à medida que a criminalidade aumentava e se tornava cada vez mais violenta, a necessidade de dar enfoque às tarefas de vigilância centradas na detecção de atividades criminosas mostrou-se essencial e os sistemas de segurança evoluíram para responder não só à presença de criminosos mas também aos tipos específicos de delitos que estavam se tornando mais comuns [Gillis 1989] [Zambon 2012].

1.1 Monitoramento

O monitoramento por câmeras tornou-se uma ferramenta essencial para garantir a segurança em diversos ambientes, como empresas, residências e locais públicos. Em estabelecimentos comerciais, por exemplo, o uso de câmeras de segurança visa prevenir furtos e assegurar a proteção de clientes e funcionários [Moreira, Sales e Moreira 2024]. Além disso uma forma de monitoramento proposta por meio do uso do Sistema de Monitoramento Inteligente baseado em visão computacional e técnicas de *Deep Learning*, facilita a identificação de armas de fogo presentes em imagens ou vídeos de segurança, a fim de reduzir o tempo em que uma ação adequada deve ser tomada [Silva 2020].

Em uma análise publicada por Baculi *et al.* (2021), na *Revista Brasileira de Economia* [Baculi et al. 2021] constatou-se uma relação direta entre homicídios e a posse de armas de fogo legais nas cinco regiões brasileiras. Ademais, pesquisas apontam que a resposta rápida das autoridades é crucial para reduzir o número de vítimas em situações que envolvem armas de fogo [Kanehisa 2018]. Portanto, a detecção ágil da presença desse tipo de armamento pode impactar diretamente na diminuição dos danos e das consequências decorrentes dessas ações.

Torna-se, portanto, necessário adotar uma abordagem automatizada para esse problema, buscando uma solução que não apenas minimize as omissões que possam ocorrer, mas também reduza o tempo de resposta diante de situações críticas. No entanto, é preciso considerar que a implementação dessas soluções pode envolver custos elevados, tanto em infraestrutura computacional quanto em treinamento e manutenção dos sistemas. Essas limitações ressaltam a importância de calibrar e aperfeiçoar de forma contínua os modelos empregados, bem como de adotar estratégias complementares que garantam que eventos relevantes não sejam

ignorados. Assim, busca-se equilibrar desempenho, precisão e viabilidade financeira na aplicação dessas tecnologias. Nesse sentido, ferramentas como o YOLO foram escolhidas para este trabalho devido à sua eficiência e rapidez na detecção de objetos em tempo real [Gu et al. 2018].

1.1.1 Lacunas no monitoramento.

Os sistemas tradicionais de circuito fechado de televisão envolvem a participação de pessoas dedicadas ao monitoramento de imagens, porém, é impossível manter um monitoramento perfeito em todos os momentos do dia, pois o desempenho do pessoal responsável pode ser afetado e comprometido por fatores como: fadiga, falta de concentração, distrações e prestar atenção a outros eventos simultâneos [Arçari 2023]. Essas condições podem ocorrer independentemente do nível de treinamento que os operadores de monitoramento recebem e, como a capacidade de atenção dos operadores é limitada, mesmo sem a presença dos fatores descritos ela está intrinsecamente relacionada à natureza da atividade de vigilância.

No Atlas da Violência, divulgado pelo Instituto de Pesquisa Econômica Aplicada [IPEA], apresenta um crescente aumento nos crimes cometidos com armas de fogo nos últimos anos. Considerando-se a perda de atenção à medida que o tempo de vigilância aumenta e que eventos de aparecimento de arma de fogo podem ocorrer por alguns segundos, qualquer sistema de vigilância que dependa exclusivamente de alguém observando atentamente a imagem entregue pela câmera terá sua eficácia seriamente comprometida. Ao considerar apenas os efeitos, a detecção tardia terá o mesmo efeito que a não detecção [Arçari 2023].

Sendo assim, a supervisão manual requer uma atenção meticulosa, o que, com o passar do tempo, pode ocasionar a exaustão dos operadores, diminuindo o êxito na identificação de ocorrências suspeitas. Com o aumento no número de câmeras, a habilidade dos operadores em analisar, simultaneamente, transmissões de vídeo diminui, prejudicando, assim, a eficiência na detecção de atividades suspeitas em diferentes locais [Yadav, Gupta e Sharma 2023].

O progresso tecnológico acarretou uma multiplicação de sistemas de captura de vídeo, os quais exercem uma função essencial em variados aspectos da sociedade contemporânea. Essas tecnologias proporcionam não apenas uma perspectiva mais

ampla e em tempo real do ambiente, mas também disponibilizam informações fundamentais para a análise e para a tomada de decisões.

A utilização de uma aplicação de captura de vídeo é fundamental para a identificação de objetos em movimento, conforme evidenciado por [Garcia-Garcia, Bouwmans e Silva 2020], podendo ser ampliada e implementada em sistemas de segurança pública nas áreas urbanas [Zheng et al. 2006]. Essa tecnologia contribui para o monitoramento do tráfego [Long et al. 2021], a detecção de incidentes de trânsito [Garcia-Garcia, Bouwmans e Silva 2020], o gerenciamento do fluxo de veículos [Colodette 2023] e a resposta eficaz a situações de emergência [Zheng et al. 2006]. Em áreas de alta concentração de pessoas, como aeroportos e estádios, as câmeras de monitoramento exercem uma função essencial na prevenção e identificação de comportamentos suspeitos, assegurando a proteção dos cidadãos [Elharrouss, Almaadeed e Al-Maadeed 2021].

No âmbito ambiental, a gravação de vídeo é essencial para o monitoramento da fauna e da flora [Pegoraro et al. 2020], a análise de padrões migratórios [Gat et al. 2022] e a avaliação da saúde dos ecossistemas [Pegoraro et al. 2020]. Essas informações favorecem a conservação do meio ambiente e o progresso sustentável.

No campo da saúde, câmeras de monitoramento exercem uma função crucial em salas cirúrgicas [Wagner et al. 2021] e em unidades de terapia intensiva [Lyra et al. 2021]. A gravação de vídeo possibilita a supervisão detalhada de práticas médicas, a investigação científica e a formação de profissionais da saúde.

Ademais, sistemas de câmeras de segurança são extensivamente empregados em contextos residenciais e comerciais [Long et al. 2021], proporcionando uma proteção suplementar, desestimulando atividades ilícitas e fornecendo provas relevantes em situações de ocorrências. Estes sistemas contemporâneos viabilizam a supervisão à distância, conferindo aos proprietários um controle ampliado sobre seus ambientes.

1.2 Principais objetivos

O principal objetivo desta pesquisa foi apresentar uma metodologia de identificação de objetos em sistemas de segurança, seja residencial, predial ou

empresarial, utilizando equipamentos e tecnologias comerciais. Sendo assim, foi utilizado um método de reconhecimento de objetos por meio do YOLO, previamente treinada com um banco de dados formado por imagens contendo os itens de interesse para a pesquisa. A partir desse ponto, o sistema iniciou o processo de captura e avaliação das informações, identificando as coordenadas das classes detectadas, além de efetuar cálculos referentes à distância e à velocidade dos objetos analisados. Nesse sentido, a pesquisa buscou demonstrar como a identificação automática de objetos pode ser utilizada comercialmente em sistemas de segurança para proteger residências, prédios e empresas. Em resumo, os objetivos iniciais da pesquisa foram:

- Treinamento de uma rede neural capaz de detectar pessoas e armas;
- Gerar um *log* com informações das classes detectadas (data/hora, classe detectada, informações das posições das caixas delimitadoras);
- Aplicar filtros para melhorar os resultados;
- Implementar a lógica *fuzzy* para determinar níveis de alertas e identificações;
- Fazer comunicação com o *bot* do Telegram para envio de alertas e *frames* capturados.

1.3 Contribuição.

A partir dos objetivos propostos na Seção 1.2, a principal contribuição desta dissertação foi a aplicação da lógica *fuzzy* para aprimorar os resultados obtidos. A integração dessa técnica permite lidar com incertezas e variações nas condições do ambiente, resultando em uma interpretação mais precisa dos dados e na redução de erros nas detecções. Esse aprimoramento contribui diretamente para o aumento da confiabilidade do sistema proposto, assegurando maior robustez e segurança nas aplicações de monitoramento e identificação de objetos. Assim, a lógica *fuzzy* não apenas complementa as abordagens tradicionais, mas se mostra essencial para elevar a qualidade e a consistência dos resultados alcançados.

1.4 Publicações Relacionadas

Este trabalho de pesquisa resultou, até a data da defesa da dissertação, em um artigo publicado na conferência *Brazilian Robotics Society* (SBRobotica2025), intitulado *Application of computer vision in surveillance cameras to identify criminal approaches.*

Este artigo apresenta uma prévia da dissertação, abordando os principais conceitos e metodologias que serão explorados ao longo desse trabalho.

1.5 Organização da Dissertação

Além desta introdução, a dissertação está organizada em mais quatro capítulos. No Capítulo 2 é apresentada a revisão teórica das principais técnicas empregadas no desenvolvimento deste trabalho, abordando um resumo dos estudos relacionados, destacando os avanços recentes em algoritmos de identificação de objetos.

O Capítulo 3 descreve a metodologia proposta para a identificação de pessoas e objetos. No Capítulo 4, são exibidos os resultados obtidos a partir da aplicação dessa visão computacional, evidenciando a eficiência da abordagem desenvolvida. Por fim, o Capítulo 5 traz as considerações finais e sugestões para futuros desdobramentos da pesquisa.

1.6 Conclusão.

Diante do que foi apresentado, verifica-se que o crescimento urbano acelerado, aliado ao aumento da criminalidade e à popularização das armas de fogo, exige soluções mais eficientes de monitoramento. A análise abordou como a evolução dos sistemas de vigilância, apoiada por tecnologias de visão computacional, pode minimizar as limitações dos métodos tradicionais.

Assim, reforça-se a relevância de integrar algoritmos robustos, como o YOLO, e métodos complementares, como a lógica *fuzzy*, para aprimorar a detecção de objetos críticos em cenários de risco. Estes conceitos fundamentam a metodologia proposta no próximo capítulo, que detalhará o desenvolvimento e a aplicação

prática do sistema de identificação automática, visando contribuir para ambientes mais seguros e para um monitoramento mais eficaz.

2 Referencial Teórico

2.1 Trabalhos relacionados

No estudo [Qi et al. 2021], aborda-se a questão de criar algoritmos de deep learning destinados à identificação automática de armas em imagens provenientes de câmeras de vigilância ou de câmeras IP inteligentes, com o propósito de enviar alertas em tempo real para as equipes de segurança. Contudo, uma das principais dificuldades na elaboração de algoritmos para a detecção de armamentos é a carência de bancos de dados públicos em grande escala, circunstância que restringe o progresso e a uniformização das investigações nesse campo. Assim, os autores sugeriram um conjunto de dados composto por 51 mil imagens etiquetadas de armas de fogo voltadas para a tarefa de detecção, além de outras 51 mil imagens recortadas, destinadas à classificação de armamentos. Um sistema para detecção de armas foi criado, o qual emprega câmeras IP inteligentes como dispositivos de bordo (edge devices) e um servidor na nuvem para a administração de dispositivos, dados e alertas, além de exercer papel crucial na diminuição da taxa de falsos positivos. A arquitetura híbrida, que integra processamento em borda e na nuvem, viabiliza tanto a detecção em tempo real quanto a classificação das armas, unindo agilidade e confiança.

Em outro trabalho, realizado por Verma [Verma e Dhillon 2017] representou um ponto importante no estudo sobre a detecção de armas de fogo utilizando técnicas de deep learning, ao evidenciar que arquiteturas fundamentadas em R-CNN(Regions with Convolutional Neural Network) poderiam ser empregadas de maneira eficaz nesse campo particular. O Faster R-CNN, ao incorporar a Rede de Propostas de Regiões (Region Proposal Network – RPN) ao procedimento de classificação, diminui consideravelmente o tempo de processamento se comparado a técnicas anteriores. No entanto, o modelo ainda apresenta limitações práticas: requer maior capacidade de processamento, possui alta latência para aplicações em tempo real e revela fragilidades em situações de oclusão parcial, mudanças abruptas na iluminação e aumento na complexidade visual do ambiente. Diferentemente do modelo de duas fases do Faster R-CNN, o YOLOv8 utiliza uma abordagem unifásica, proporcionando, desse modo, uma maior velocidade na inferência, sem

comprometer a precisão. Além disso, a sua estrutura, bem como a implementação de técnicas e ajustes nas funções de perda, conferem ao modelo uma maior robustez para lidar com diferentes escalas, ângulos e condições ambientais. Essa evolução confere ao YOLO um desempenho adequado em aplicações práticas, incluindo sistemas embarcados e monitoramento em tempo real. A análise comparativa realizada entre os dois modelos enfatiza o progresso das arquiteturas de detecção de objetos e justifica a opção pelo YOLOv8 neste estudo, considerando tanto os avanços tecnológicos quanto a viabilidade de sua aplicação em sistemas que exigem respostas ágeis e precisas.

Já para o trabalho de [Deshpande et al. 2023] há uma validação experimental para a aplicabilidade do YOLOv8s no fortalecimento de operações militares, evidenciando seu potencial como um verdadeiro multiplicador de forças em cenários de combate. A pesquisa demonstra a adaptabilidade do modelo frente a condições ambientais variáveis, fator essencial em aplicações militares no mundo real. Os resultados apresentados mostraram que a rede mantém desempenho consistente em diferentes terrenos, condições de iluminação e situações climáticas, o que amplia significativamente sua viabilidade operacional. Essa capacidade de adaptação assegura uma detecção confiável de armamentos mesmo em cenários adversos, reforçando sua relevância para a segurança e para o apoio tático.

Em seu estudo [Bustamante e Gutiérrez 2024], aborda a necessidade de se aplicar medidas de segurança mais eficazes em um país da América Latina, em que pelo menos 25% dos moradores das principais cidades já foram vítimas de crimes. É apresentada uma abordagem inovadora para reforçar a segurança pública por meio da utilização de redes neurais baseadas em DETR para a detecção de armas de fogo em tempo real por câmeras de vigilância. Foi explorada a capacidade do RT-DETR, para a detecção de objetos em tempo real sem comprometer a precisão. O modelo, avaliado no dispositivo embarcado Nvidia Jetson AGX Xavier, alcançou um notável F1-score de 99,52% a 38 quadros por segundo.

No estudo de [Kundeti 2025] , foi proposto uma melhoria da segurança pública por meio da integração de sistemas avançados de detecção de armas de fogo. Foi feita uma análise comparativa entre duas das principais tecnologias em técnicas de detecção de objetos, o RT-DETR e o YOLOv8. Teve como objetivo demonstrar

qual modelo apresenta melhor desempenho em termos de precisão, robustez e adaptabilidade na detecção de armas curtas em tempo real, especialmente em espaços públicos. Além disso, os modelos foram treinados com o mesmo banco de dados, formado por imagens de armas de fogo. Os resultados foram avaliados com base nas métricas de mAP, *precision* e *recall* para diferentes classes. O RT-DETR teve um desempenho ligeiramente superior em todas as métricas, enquanto o YOLOv8 demonstrou ser mais estável e robusto frente a diferentes configurações de parâmetros.

Em seu trabalho [Soares et al. 2024] buscou aplicar múltiplas redes neurais em conjunto com a DETR na detecção de armas de fogo em imagens de vigilância, com o objetivo de melhorar a vigilância remota, utilizando inserção de redes neurais profundas. A metodologia que foi proposta articulava a colaboração e a autocoordenação das redes nas camadas full connected da DETR por meio da técnica de múltiplas redes neurais artificiais (MRNA) autocoordenadas, que dispensa o uso de coordenador. Essa autocoordenação consiste em fazer com que as redes sejam treinadas, uma após a outra, e as suas saídas sejam integradas sem um elemento extra chamado de coordenador. O trabalho de [Soares et al. 2024] é o primeiro a introduzir a MRNA autocoordendas em arquiteturas de detecção de objetos baseadas em transformadores para a detecção de objetos.

Com o intuito de reduzir a carga de trabalho de operadores de câmeras de vigilância, [Nguyen et al. 2024] fez um experimento com o método de aprendizado por transferência utilizando o modelo DETR,e pode, mesmo utilizando um banco de dados reduzido, conseguiu fazer com que a rede convergisse rapidamente.

Com relação ao trabalho de [Jain, Aishwarya e Garg 2020], é aplicada a detecção de objetos baseada em um sistema em tempo real para detecção de armas, com reconhecimento do tipo e modelo. A abordagem utiliza vídeo em tempo real como entrada, empregando o classificador Haar Cascade, em conjunto com a biblioteca OpenCV. Foi desenvolvido com foco em aplicações voltadas à gestão da segurança e proteção, sendo capaz de identificar e classificar armas em vídeos com alta taxa de acurácia — alcançando até 95% de precisão na detecção de submetralhadoras.

2.2 ROS - Robot Operating System

O ROS é um *framework* amplamente empregado no desenvolvimento de *software* voltado para a robótica e que disponibiliza um conjunto de ferramentas, bibliotecas e pacotes que possibilitam a programação e o controle eficaz de robôs, além de promover a integração de diversos sensores, atuadores e algoritmos. Embora seu nome sugira o contrário, o ROS não se configura como um sistema operacional integral, mas sim como uma camada que funciona sobre o Linux [Macenski et al. 2022].

Uma ferramenta de extrema relevância na análise de sistemas robóticos é o rqt_graph, um recurso visual que integra o ROS e possibilita a visualização da estrutura de comunicação entre os nós e tópicos de um sistema em funcionamento. Ele gera um gráfico que representa os nós (processos) como blocos e os tópicos (fluxos de dados) como conexões entre esses blocos. Tal representação é de fundamental importância para compreender, diagnosticar e investigar as interações entre os diversos componentes do sistema [Li, Hasegawa e Azumi 2022].

Uma das características mais significativas do ROS é a sua estrutura modular fundamentada em nós ou *nodes* (processos responsáveis pela execução de tarefas específicas) e tópicos (canais de comunicação que interligam os nós). Isso possibilita que os elementos de um sistema robótico, como câmeras, motores, sensores de proximidade e algoritmos de controle, compartilhem informações de forma eficiente e em tempo real. Adicionalmente, o ROS é um *framework* de código aberto, o que permite uma expressiva colaboração da comunidade global de robótica, contando com milhares de pacotes já elaborados e disponíveis para implementação em projetos [Li, Hasegawa e Azumi 2022].

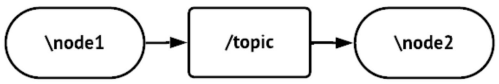
A configuração de nós e tópicos é essencial para a modularidade e adaptabilidade do ROS, possibilitando que diversos componentes de um robô se intercomuniquem de forma eficaz e independente. Um nó no ROS representa um processo singular que realiza uma função específica. Um nó pode, por exemplo, ter a função de capturar imagens de uma câmera, enquanto outro é incumbido do processamento dessas imagens para identificar objetos. Esses nós não se comunicam de forma direta, mas realizam a troca de informações por meio de tópicos, os quais

funcionam como canais de comunicação [Macenski et al. 2022].

Um tópico consiste em um fluxo de dados que é designado por um nome específico. Um nó tem a capacidade de publicar informações em um tópico, enquanto outros nós podem se registrar para obter esses dados. Esse modelo é denominado *publish-subscribe* e assegura que a comunicação seja tanto flexível quanto escalável, possibilitando que diversos nós publiquem ou se inscrevam em um mesmo tópico, conforme a demanda [Mayoral-Vilches e Corradi 2021].

De forma simples e ilustrativa, será feita uma breve explicação da forma como se dá a comunicação entre nós por um tópico, como exemplificado na Figura 1. Considere que há dois nós:

Figura 1 – Exemplo do rqt_graph de dois nós e um tópico.



Fonte: Elaborado pelo autor.

- Nó 1: Divulga as informações de temperatura de um sensor.
- Nó 2: Recebe essas informações e as analisa para determinar se a temperatura se encontra em um intervalo adequado.

A comunicação ocorre da seguinte maneira:

- Nó 1 publica um assunto: O Nó 1, denominado, por exemplo, de 'sensor_node', realiza a leitura da temperatura de um sensor e dissemina essas informações em um tópico intitulado *temperature*. Esse nó estabelece o tópico, determina a periodicidade na qual os dados são divulgados (como, por exemplo, uma vez por segundo) e transmite as mensagens.
- Nó 2 registra-se no tópico: O Nó 2, denominado 'monitor_node', deve receber os dados de temperatura que são publicados pelo Nó 1. Para tal, ele realiza a inscrição no tópico *temperature*. Sempre que uma nova mensagem for

veiculada neste tópico, o Nó 2 será informado e receberá as informações para processamento [Macenski et al. 2022].

No ROS, as mensagens são formatadas conforme tipos específicos de mensagem, como 'std_msgs/Float64', que representa um valor numérico de ponto flutuante. Neste contexto, o Nó 1 emite mensagens do tipo 'std_msgs/Float64', enquanto o Nó 2 aguarda a recepção de mensagens dessa mesma categoria [Mayoral-Vilches e Corradi 2021].

Essa arquitetura, fundamentada na estruturação em nós e tópicos, proporciona uma variedade de vantagens para a criação de sistemas modulares e eficazes no ROS. Um dos aspectos mais relevantes é o desacoplamento entre os componentes. Os nós não necessitam ter conhecimento dos pormenores operacionais uns dos outros, mas apenas dos assuntos pertinentes à sua comunicação. Essa particularidade facilita o aperfeiçoamento, a troca ou a atualização de componentes isolados, sem afetar o restante do sistema [Mayoral-Vilches e Corradi 2021].

Um benefício relevante é a escalabilidade. Uma vez que múltiplos nós têm a capacidade de se inscrever em um mesmo tópico ou divulgar informações em tópicos diferentes, torna-se possível criar sistemas altamente complexos sem aumentar substancialmente a complexidade estrutural. Esse modelo de comunicação possibilita a incorporação de novas funcionalidades ou a ampliação de capacidades, sem que o sistema se torne complicado de gerir [Macenski et al. 2022].

Por fim, esse *framework* oferece grande flexibilidade, possibilitando alterações simplificadas no sistema. É possível adicionar novos nós de forma simples, assim como tópicos podem ser modificados, criados ou redirecionados, sem a exigência de redesenhar toda a estrutura. Essa capacidade de adaptação é particularmente benéfica em aplicações que demandam experimentação ou modificações recorrentes, tal como no desenvolvimento de robôs ou em sistemas de visão computacional. Essas características tornam o modelo de nós e tópicos do ROS uma opção apropriada para aplicações que requerem robustez, escalabilidade e modularidade [Li, Hasegawa e Azumi 2022].

O ROS é extensivamente empregado em uma variedade de aplicações na área de robótica. Por exemplo, sua importância é fundamental no progresso de veículos

autônomos, nos quais distintos subsistemas, tais como percepção, planejamento e controle, devem ser integrados. Além disso, é extensivamente utilizado em robôs móveis, manipuladores robóticos industriais, drones e sistemas de teleoperação em condições adversas, como na exploração espacial ou submarina. Outro exemplo significativo são os robôs de serviço, os quais empregam o ROS para atividades de mapeamento, navegação e interação com seres humanos [Li, Hasegawa e Azumi 2022].

Um dos principais fatores que fazem com que o ROS seja amplamente utilizado para a avaliação de novos dispositivos é a sua habilidade de integração com simuladores como Gazebo e RViz. Esses simuladores possibilitam que programadores realizem testes de sensores, atuadores e algoritmos em ambientes virtuais antes de sua implementação em robôs físicos. Tal abordagem diminui despesas e perigos ao longo do procedimento de desenvolvimento. Ademais, o ROS disponibiliza suporte a linguagens de programação como *Python* e C++, podendo ser aplicado a uma variedade de *hardwares*, com o suporte de pacotes como o CUDA, para placas gráficas que possibilitam um desempenho aprimorado em aplicações que envolvem o processamento de imagens e vídeos [Macenski et al. 2022].

O processo de trabalho com ROS geralmente se inicia com a configuração dos nós e tópicos, a fim de gerenciar os diversos dispositivos do robô, como câmeras e motores. A partir desse ponto, algoritmos são desenvolvidos para funções específicas, como a manipulação de imagens, o mapeamento ou a regulação de movimento. No decorrer do desenvolvimento, instrumentos de depuração e visualização, como o RViz, possibilitam a supervisão do funcionamento do sistema, além de permitir a rápida identificação de problemas. Uma vez que todos os componentes estejam operando com a simulação, o código pode ser transferido para o robô físico para a realização de testes, sendo controlado de maneira remota pelo *framework* [Li, Hasegawa e Azumi 2022].

Devido à sua flexibilidade, extensibilidade e ampla comunidade, o ROS é reconhecido como um padrão no desenvolvimento de robôs, promovendo o avanço na pesquisa e na indústria robótica ao oferecer uma plataforma acessível e robusta para experimentação e inovação [Mayoral-Vilches e Corradi 2021].

2.3 Visão computacional e OpenCV

A visão computacional é um campo da inteligência artificial que capacita máquinas a interpretarem e processarem informações visuais do mundo, permitindo que executem tarefas como reconhecimento de objetos, análise de imagens e vídeos e tomada de decisões baseada em dados visuais. Além disso, é um campo de estudo vasto com diversas aplicações práticas e teóricas [Milano e Honorato 2014].

Uma das bibliotecas mais usadas é a OpenCV, que, por ser um código aberto, é amplamente empregada em tarefas de processamento de imagens e visão computacional. Inserida no mercado em 2000, essa ferramenta rapidamente se tornou uma das mais utilizadas por desenvolvedores e pesquisadores que atuam na análise de imagens e vídeos. O OpenCV disponibiliza uma ampla variedade de recursos, abrangendo processamento de imagens, detecção e acompanhamento de objetos, reconhecimento facial, calibração de câmeras, reconstrução tridimensional e análise de movimento. Sua eficácia e adaptabilidade possibilitam sua aplicação em várias linguagens de programação, tais como C++, *Python* e Java, e em diversas plataformas, incluindo dispositivos móveis [Gulati et al. 2022].

O OpenCV teve uma grande importância na evolução de tecnologias e investigações relacionadas à visão computacional. Devido ao seu código aberto e à sua acessibilidade, possibilitou a democratização do acesso a ferramentas sofisticadas de processamento de imagens, o que permitiu que pesquisadores e engenheiros criassem soluções inovadoras. Tecnologias como o reconhecimento facial em dispositivos móveis, a detecção de pedestres em veículos autônomos, sistemas de vigilância inteligentes e até mesmo a realidade aumentada foram diretamente beneficiados pela versatilidade do OpenCV [Brahmbhatt 2013].

A conjugação do OpenCV com o *Robot Operating System* (ROS) é significativa para a elaboração de inovações tecnológicas. Devido à sua característica modular, o ROS propicia uma comunicação eficiente entre componentes distintos de um sistema robótico, como sensores, atuadores e algoritmos de controle. A combinação do OpenCV com o ROS possibilita que sistemas robóticos executem atividades de visão computacional, incluindo navegação orientada pela visão, manipulação de objetos e interação entre humanos e robôs. A biblioteca OpenCV

realiza o processamento dos dados visuais obtidos por meio de sensores e câmeras, ao passo que o ROS é responsável por gerenciar a comunicação e a coordenação entre os diferentes módulos. A referida integração propicia o avanço de sistemas sólidos e escaláveis, abrangendo aplicações que variam desde automóveis autônomos até drones e robôs industriais [Alberola, Gallego e Maestre 2019].

Adicionalmente, o OpenCV impulsionou a investigação em domínios como *Machine Learning* e *Deep Learning* ao disponibilizar suporte para a integração com frameworks como *TensorFlow* e *PyTorch*. Tal associação torna mais eficiente o treinamento e a implementação de modelos de aprendizado em atividades como segmentação semântica, classificação de imagens e detecção de objetos em tempo real [Harrison 2019].

A integração do OpenCV com o ROS distingue-se por sua habilidade de converter concepções em protótipos operacionais de forma ágil e eficaz. Utilizando simuladores como o Gazebo e ferramentas de visualização como o RViz, pesquisadores têm a possibilidade de testar e validar algoritmos de visão computacional antes de sua implementação em sistemas físicos. Tal abordagem diminui os custos, os riscos e o tempo de desenvolvimento, convertendo o OpenCV e o ROS em ferramentas essenciais para a inovação na robótica. Em um universo progressivamente automatizado, essa interação continua a promover avanços significativos em várias indústrias e na pesquisa científica [Silva, Chaves e Aquino 2012].

A aplicação de visão computacional funciona da seguinte forma:

- Inicialmente há a captura da imagem, realizada por sensores ou câmeras;
- depois há o pré-processamento, que ajusta a qualidade das imagens removendo ruídos e realçando características importantes, como bordas e texturas; e,
- por fim, ocorre a análise, na qual algoritmos e redes neurais realizam tarefas como segmentação, classificação e detecção de objetos.

A detecção de objetos é uma das áreas centrais da visão computacional. Ela envolve a identificação e a localização de objetos específicos dentro de uma imagem ou sequência de vídeo, fornecendo informações essenciais para aplicações que exigem compreensão detalhada do ambiente visual [Ribeiro et al. 2024]. Uma das formas

de aplicar a detecção de objetos envolve a criação de caixas delimitadoras em torno de objetos previamente identificados [Arnold 2016]. Essas caixas delimitadoras são usadas para determinar a posição exata de objetos em uma determinada cena ou para rastrear seu movimento dentro dela [Oliva 2024].

O papel da detecção de objetos na Visão Computacional vai muito além da identificação de objetos, sendo este um mecanismo essencial para entender contextos visuais complexos. Essa tecnologia permite tarefas diferenciadas, como distinguir instâncias de objetos individuais (segmentação de instâncias), entender cenas para gerar texto descritivo (adicionar legendas às imagens) e detecção e rastreamento contínuos de objetos em tempo real em sequências de vídeo [Oliva 2024] [Arnold 2016]. Além disso, suas aplicações se estenderam a vários campos, desde a melhoria da segurança pública por meio da detecção e rastreamento de pedestres e veículos, até a transformação do setor varejista com caixas que permitem pagamentos automatizados, sem a necessidade de escanear cada item individualmente [SICSÚ 2023].

Avanços em aprendizado de máquina, modelos de aprendizado profundo e redes neurais levaram a detecção de objetos a novos patamares, permitindo o processamento em tempo real e a alta precisão, importantes para ambientes dinâmicos, como direção autônoma ou sistemas avançados de vigilância [Ludermir 2021].

Esses avanços ressaltam o impacto transformador da detecção de objetos tanto nos avanços técnicos quanto na vida cotidiana. O propósito é treinar um programa de computador para reconhecer diferentes tipos de objetos, detectá-los e contá-los e, então, localizar automaticamente os objetos em sua posição precisa no *pixel* mais próximo em novas imagens [Rosa et al. 2023]. Para tal, o sistema é alimentado por milhares de fotos anotadas, nas quais cada objeto de interesse é identificado por uma caixa delimitadora (Arnold, 2016).

Diversas arquiteturas são empregadas nesse setor. Além do YOLO, Faster R-CNN [Maity, Banerjee e Chaudhuri 2021], SSD [Zhai et al. 2020], DETR (Detection Transformer) e RF-DETR(Real-Time Detection Transformer) [Robicheaux et al. 2025], que são amplamente utilizados na detecção de objetos, destaca-se também o Haar Cascade [Soo 2014], reconhecido como um método tradicional fundamentado

em aprendizado de máquina voltado para o reconhecimento facial e de objetos simples e o Grounding DINO que adiciona um contexto semântico na identificação dos obejtos [Liu et al. 2023]. Sistemas modernos adicionais englobam o Mask R-CNN [He et al. 2017], que efetua a segmentação de objetos, e a *EfficientDet* [Tan, Pang e Le 2020] desenvolvida para proporcionar maior eficiência em dispositivos com recursos restritos.

2.4 DETR

A *Detect transformer*, refere-se a um modelo que emprega mecanismos de atenção para o processamento de dados. Ao contrário das redes neurais recorrentes (RNNs) ou convolucionais (CNNs), que processam sequências ou imagens analisando elementos de maneira local ou em etapas, os transformadores têm a capacidade de considerar todas as partes de uma entrada simultaneamente, estabelecendo ligações diretas entre elas. Na prática, isso implica que o modelo é capaz de reconhecer dependências de longo alcance em uma sequência textual ou em uma imagem. Na área da visão computacional, o transformador é capaz de identificar que dois objetos localizados em lados opostos da imagem podem estar interconectados, uma tarefa que uma rede exclusivamente convolucional teria maior dificuldade em discernir [Shehzadi et al. 2023].

As principais características que definem um transformador são:

- Autoatenção: cada elemento da entrada "presta atenção" a todos os outros, atribuindo pesos de relevância.
- Paralelismo: diferentemente das RNNs, os transformadores processam dados em paralelo, o que aumenta a eficiência.
- Generalidade: a mesma estrutura foi aplicada com sucesso em processamento de linguagem natural (como em modelos de tradução automática), em visão computacional, e até em áudio e multimodalidade.

As CNNs não se beneficiam de forma tão intensa do pré-treinamento em larga escala, podendo apresentar uma eficácia inferior ou convergir de maneira mais lenta. Em diversas outras áreas do aprendizado de máquina, desde a classificação

de imagens até os modelos de linguagem de grande escala, o pré-treinamento torna-se cada vez mais crucial para a obtenção de resultados significativos. Portanto, um detector de objetos que apresenta uma significativa vantagem proveniente do pré-treinamento é propenso a proporcionar melhores resultados na detecção de objetos [Shehzadi et al. 2023].

Recentemente, com a introdução do RT-DETR em 2023, demonstrou-se que a família de modelos DETR equipara-se aos YOLO em termos de latência, que é uma etapa de pós-processamento necessária para os modelos YOLO, mas dispensável para os DETR. Além disso, tem-se realizado um considerável volume de trabalho para possibilitar a rápida convergência dos DETRs [Shehzadi et al. 2023].

Avanços recentes em DETRs combinam esses dois fatores para desenvolver modelos que, na ausência de pré-treinamento, igualam o desempenho de YOLOs e, com o pré-treinamento, superam significativamente em uma determinada latência. Há também motivos para crer que um pré-treinamento mais robusto potencializa a capacidade de um modelo em aprender a partir de pequenas quantidades de dados, o que é de extrema relevância para tarefas que podem não contar com conjuntos de dados de escala COCO. Abordagens híbridas também estão em processo de evolução. O YOLO-S combina transformadores e redes neurais convolucionais (CNNs) para oferecer desempenho em tempo real. Algumas distribuições mais recentes do YOLO também utilizam da aprendizagem sequencial em conjunto com transformadores.

2.5 GroundingDINO

O Grounding DINO constitui um avanço considerável no campo da detecção de objetos, ao combinar princípios dos modelos de detecção fundamentados em transformadores com noções de alinhamento semântico entre a linguagem e a percepção visual. Criado com base no DINO (DETR com Caixas de Âncora Melhoradas para DeNoising), este modelo possui a funcionalidade de executar detecção orientada pela linguagem natural, ou seja, é capaz de identificar e localizar objetos em uma imagem a partir de descrições textuais apresentadas pelo usuário. Essa característica coloca o Grounding DINO dentro do grupo de abordagens conhe-

cidas como modelos de visão-linguagem, que se destacam por integrar informações oriundas de diferentes modalidades em uma representação compartilhada [Liu et al. 2023].

Sob a perspectiva técnica, o Grounding DINO emprega uma estrutura de base robusta para a extração de características visuais, acompanhada por um módulo de transformadores que realiza o processamento simultâneo dos tokens correspondentes à imagem e ao texto. O entendimento entre essas duas modalidades ocorre por meio de mecanismos de atenção cruzada, os quais possibilitam ao modelo relacionar áreas específicas da imagem às palavras ou expressões textuais equivalentes. Essa perspectiva possibilita que a detecção seja orientada por diretrizes mais adaptáveis e descritivas, contrastando com as estruturas convencionais que operam unicamente com um conjunto fixo de categorias preestabelecidas. Dessa forma, o modelo apresenta a habilidade de reconhecer objetos pertencentes a categorias que não foram previamente incorporadas em seu treinamento, desde que exista uma descrição textual apropriada, aumentando substancialmente sua capacidade de generalização [Liu et al. 2023].

No âmbito da identificação de objetos, o Grounding DINO representa um avanço significativo. Ao passo que modelos tradicionais, como o YOLOv8 ou diversas variações da família DETR, estão restritos a um conjunto fixo de classes previamente anotadas, o Grounding DINO possibilita que o procedimento de detecção seja dinâmico e guiado pela linguagem. Essa questão é especialmente significativa em contextos que demandam flexibilidade, como sistemas de busca multimodal, robótica interativa, veículos autônomos e monitoramento inteligente, em que tanto o ambiente quanto o interesse do usuário podem sofrer alterações em tempo real [Son e Jung 2024].

No que diz respeito ao desempenho, o Grounding DINO tem mostrado ser competitivo em relação aos principais detectores em benchmarks convencionais, ao passo que proporciona a benefício de um vocabulário amplo para a detecção.

Assim, no contexto presente da identificação de objetos, o Grounding DINO representa a união entre a detecção visual de elevada performance e a interpretação semântica da linguagem. Ao permitir que a linguagem natural funcione como referência para a detecção, o modelo expande o escopo das arquiteturas de detecção,

estabelecendo uma interação mais intuitiva com os usuários e promovendo aplicações mais dinâmicas na realidade.

2.6 YOLO

O YOLO é um dos pacotes mais conhecidos e eficientes para a identificação de objetos em tempo real. Foi elaborado com a finalidade de integrar elevada precisão na identificação de objetos a um desempenho extremamente ágil. Historicamente, os modelos YOLO baseados em CNN apresentaram a melhor precisão entre os detectores de objetos em tempo real. As redes neurais convolucionais permanecem sendo um elemento essencial em várias das melhores abordagens na visão computacional, por exemplo, A abordagem do D-FINE utiliza tanto redes neurais convolucionais (CNNs) quanto transformadores.

Trata-se de uma estratégia fundamentada em redes neurais convolucionais (CNNs) destinada à identificação e localização de objetos em imagens ou vídeos. Ao contrário dos métodos convencionais que fragmentam a detecção em diversas fases, como a extração de características e a classificação, o YOLO utiliza uma abordagem integrada. Ele realiza o processamento da imagem de forma integral em uma única etapa, ao examinar todas as áreas de maneira simultânea. Essa característica o distingue de outros métodos, como o Faster R-CNN, que efetuam a detecção em etapas distintas [Gomes 2022].

Segundo [Jocher, Chaurasia e Qiu 2023], o funcionamento do YOLO pode ser dividido em algumas etapas principais:

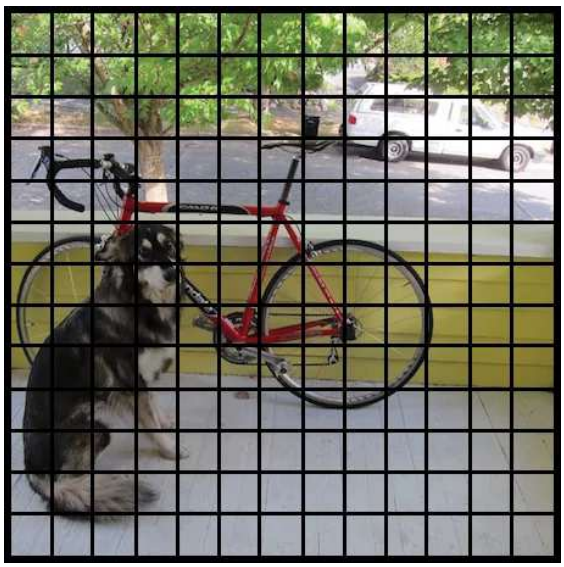
- Divisão da Imagem em Grades: A imagem de entrada é dividida em uma grade (por exemplo, 13x13 ou 19x19 células). Cada célula da grade é responsável por prever caixas delimitadoras, os *bounding boxes*, e a classe do objeto presente dentro dessa célula.
- Predição de Caixas e Classes: para cada célula o YOLO prevê:
  - A posição e o tamanho das caixas delimitadoras (x, y, largura e altura).
  - A confiança de que a caixa contém um objeto.

  - A probabilidade das classes possíveis para os objetos detectados (por exemplo, ‘carro’, ‘pessoa’ ou ‘cachorro’).

- *Non-Maximum Suppression* (NMS): técnica utilizada pelo YOLO para evitar redundâncias ao eliminar caixas sobrepostas e manter apenas as mais confiáveis para cada objeto detectado.

- Processamento Único: diferentemente de métodos tradicionais que analisam múltiplas regiões da imagem em etapas separadas, o YOLO processa toda a imagem de uma só vez, o que reduz significativamente o tempo de inferência.

Figura 2 – Grid feito na imagem base com objetos a serem identificados.



Fonte: Retirado de [Balasubramanian et al. 2019].

O primeiro procedimento executado pelo algoritmo YOLO envolve a segmentação da imagem em uma grade de S×S células (Figura 2). Nas primeiras versões do modelo, essa grade é composta por dimensões de 13x13, o que totaliza 169 células, porém, nas edições mais recentes do YOLO, essa grade foi expandida para 19x19 [Costa, Oliveira e Mariano 2023].

Cada unidade da grade é encarregada de antecipar a existência de até cinco caixas delimitadoras, uma vez que muitos objetos podem estar situados em

uma mesma unidade. Cada caixa delimitadora examina uma fração da imagem, capturando dados significativos acerca da área associada. No exemplo apresentado, a versão do YOLO possui um total de 845 caixas delimitadoras (13×13×513 × 13 × 5) [Costa, Oliveira e Mariano 2023].

A caixa delimitadora consiste no retângulo que determina a posição de um objeto na imagem. No decorrer do processo de identificação de objetos é possível encontrar múltiplas caixas delimitadoras vinculadas a um único objeto, as quais podem apresentar tamanhos diversos(Figura 3) [Jocher, Chaurasia e Qiu 2023].

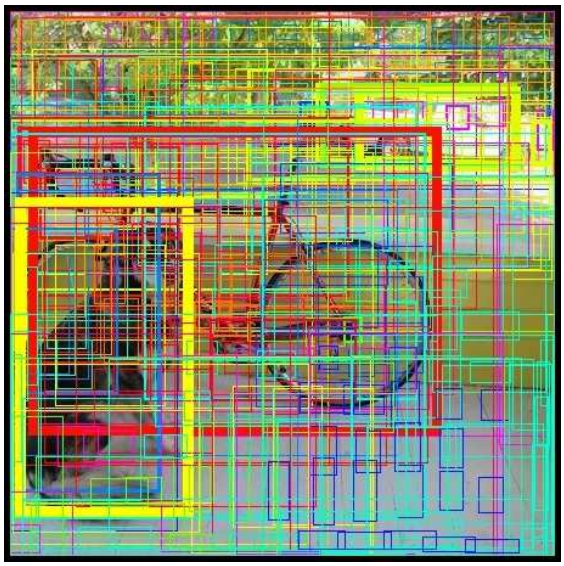
Figura 3 – *Bounding boxes* feitos a partir do grid.

Fonte: Retirado de [Balasubramanian et al. 2019].

Cada caixa delimitadora vem acompanhada de uma pontuação de confiança, a qual indica a probabilidade de que um objeto esteja realmente localizado dentro dessa caixa. Entretanto, essa pontuação não especifica a natureza do objeto identificado. Na Figura 3, as caixas identificadas pelo YOLO foram evidenciadas e a espessura das bordas das caixas reflete o grau de confiança do algoritmo na existência de um objeto: quanto mais elevada a confiança, mais grossa será a borda da caixa. A quantidade de objetos potenciais reconhecidos nesta etapa do algoritmo é significativa [Gomes 2022].

Além disso, o procedimento de categorização dos objetos é executado para cada caixa delimitadora identificada. Esse procedimento consiste na atribuição de uma probabilidade a cada uma das classes de objetos que o modelo foi treinado para identificar. A Figura 4 a seguir representa essa classificação, evidenciando, por meio de diferentes cores, os objetos correspondentes a cada caixa delimitadora [Costa, Oliveira e Mariano 2023].

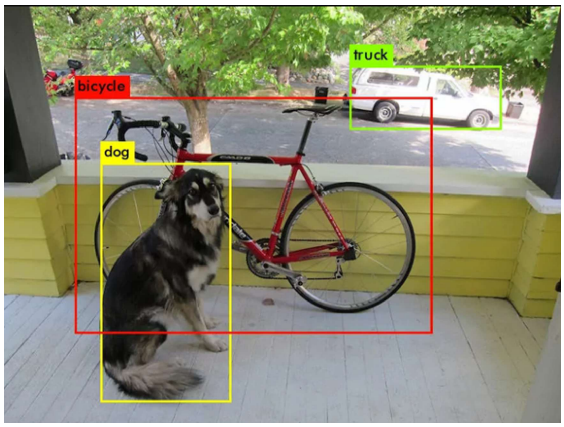
Figura 4 – *Bounding boxes* com a maior parte identificada.



Fonte: Retirado de [Balasubramanian et al. 2019].

A junção da pontuação de confiança da caixa com a probabilidade da classe gera a pontuação final, a qual expressa a probabilidade de uma caixa delimitadora abrigar um determinado objeto. É importante notar que numerosas caixas identificadas exibem uma baixa pontuação de confiança. Para excluir as caixas não pertinentes é possível estabelecer um limiar de confiança, o qual o usuário pode ajustar conforme necessário dependendo dos requisitos específicos da aplicação, desconsiderando aquelas que possuem uma pontuação inferior a esse patamar. Para o exemplo em questão, o intervalo de confiança definido foi de 30%, exibido na Figura 5, na qual é exibida a *bounding box* com o nome do objeto identificado [Jocher, Chaurasia e Qiu 2023].

Figura 5 – *Boxes* finais.



Fonte: Retirado de [Balasubramanian et al. 2019].

A rede neural do YOLO emprega dados de toda a imagem para estimar a posição de cada caixa delimitadora. Ademais, contempla, de maneira simultânea, todas as caixas delimitadoras referentes a cada classe de objetos, o que indica que a rede efetua um raciocínio abrangente acerca da imagem e dos objetos nela contidos. Portanto, o principal destaque do YOLO é sua agilidade, alcançando taxas de processamento superiores a 30 FPS (*frames* ou quadros por segundo) em equipamentos de alta performance. Em uma aplicação do YOLO com o treinamento do banco de dados é possível perceber que há a identificação de duas classes na imagem, *human-with-weapon* e *pistol*, e que em ambas há a exibição da confiabilidade da rede nos objetos detectados, conforme exibido na Figura 6 [Gomes 2022] [Jocher, Chaurasia e Qiu 2023]

Figura 6 – Exemplo da rede do trabalho operando.

Fonte: Adaptado de https://encurtador.com.br/VmP5U

Além da rapidez na identificação, essa rede também se destaca pela elevada precisão. Ela foi concebida para reduzir a incidência de erros de identificação, como detecções incorretas ou a exclusão de objetos. A estratégia de levar em conta o contexto global da imagem possibilita que o algoritmo reconheça objetos, mesmo em situações complicadas, nas quais ocorre sobreposição ou condições de iluminação adversas [Gomes 2022].

O YOLO é amplamente utilizado, conforme [Gomes 2022], em:

- Veículos autônomos: para detecção de pedestres, placas de trânsito, outros veículos e obstáculos.
- Sistemas de segurança: em câmeras de vigilância para identificar intrusos ou atividades suspeitas.
- Drones: para rastreamento de alvos, detecção de terrenos e mapeamento aéreo.
- Robótica: navegação autônoma, manipulação de objetos e interação com o ambiente.

- Reconhecimento em vídeos: como análise de cenas e rastreamento de múltiplos objetos em tempo real.

A detecção de objetos pertence a um ramo da inteligência artificial chamado Visão Computacional [Arnold 2016]. O software de detecção de objetos visa reproduzir os processos complexos da visão humana por meio de diversos dados de treinamento e da orquestração de algoritmos complexos: as máquinas compreendem o mundo visual com um nível de precisão e sofisticação antes reservado exclusivamente à percepção humana [SICSÚ 2023].

2.6.1 Treinamento do YOLO

O treinamento do YOLO foi feito, a princípio, utilizando o Google *Colab*, uma plataforma gratuita que fornece GPUs gratuitas e de simples configuração, tornando-a uma opção ideal para treinar modelos de visão computacional. Posteriormente, foram feitos novos treinamentos à medida em que o banco de dados foi aumentando.

Segundo [Jocher, Chaurasia e Qiu 2023], o treinamento do YOLO no Google *Colab* segue etapas bem definidas:

- Preparação do Ambiente:
  - Inicialmente, configura-se o ambiente instalando as dependências necessárias, como bibliotecas específicas para o YOLO (por exemplo, *PyTorch*, *TensorFlow*, ou *frameworks* adaptados, como *Ultralytics* YOLO).
  - Além disso, verifica-se o suporte a GPU no *Colab* para acelerar o treinamento.
- Organização do Conjunto de Dados:
  - Os dados precisam ser organizados no formato esperado pelo YOLO. Isso geralmente envolve: as imagens de entrada, os arquivos de anotação que especificam a classe dos objetos e suas caixas delimitadoras (em formatos como COCO ou YOLO-*specific*).

- O conjunto de dados é carregado no *Colab* por meio de *upload* direto ou integração com o *Google Drive*.
- Configuração do Modelo:
  - Define-se um arquivo de configuração especificando os hiperparâmetros do modelo, como o número de classes, o tamanho do lote (*batch size*) e o tamanho da imagem.
  - Arquivos de pesos pré-treinados (por exemplo, no COCO *dataset*) são frequentemente usados para *transfer learning*, acelerando o treinamento.
- Execução do Treinamento:
  - Durante o treinamento a rede neural ajusta seus pesos com base no conjunto de treinamento, minimizando a função de perda a cada época (*epoch*).
  - Durante o treinamento, o modelo gera saídas como a perda total, as perdas individuais (localização, confiança e classificação) e as métricas de validação.
- Avaliação e Exportação:
  - Após o treinamento, o modelo treinado é avaliado em um conjunto de teste para verificar sua precisão e capacidade de generalização.
  - Por fim, os pesos finais do modelo são salvos e exportados para uso em aplicações reais.

O treinamento do YOLO gera várias saídas úteis para avaliar o progresso e o desempenho do modelo:

- Funções de Perda (*Loss*): Durante cada *epoch* a perda total é reportada, incluindo subcomponentes como:
  - Perda de localização (erros nas caixas delimitadoras).

  - Perda de confiança (erros na probabilidade de que uma caixa contenha um objeto).
  - Perda de classificação (erros na classe do objeto detectado).

- Mean Average Precision (mAP): Métrica que avalia a precisão do modelo em todas as classes e limiares de confiança.
- Pesos do Modelo: Pesos intermediários (salvos em *checkpoints*) e finais são exportados para serem usados no modelo treinado.
- Gráficos de Métricas: Visualizações da perda ao longo do tempo e do desempenho em dados de validação.

Uma variável significativa a ser estabelecida durante o treinamento, que afeta diretamente a duração deste processo, é o número de *epochs* que devem ser analisadas, pois este representa a quantidade de vezes que o modelo passa por todo o conjunto de dados de treinamento [Gomes 2019].

A quantidade de *epochs* durante o treinamento do YOLO apresenta consequências relevantes, tanto favoráveis quanto desfavoráveis. De maneira positiva, um aumento no número de *epochs* proporciona ao modelo um maior número de oportunidades para ajustar seus pesos, aprimorando assim sua capacidade de identificar padrões mais sofisticados nos dados. Dessa forma, é possível aprimorar a precisão, adequando-se de maneira mais eficaz aos pormenores característicos de cada classe de objeto. Ademais, em determinadas situações, uma quantidade adequada de *epochs* pode propiciar uma generalização mais eficiente. Ao ser submetido a um maior número de ciclos de treinamento o modelo pode adquirir representações mais sólidas e generalizáveis, o que é essencial para que o desempenho permaneça estável em relação a dados que não foram apresentados durante o processo de treinamento [Araújo et al. 2022].

Em contrapartida, o treinamento em demasia pode ocasionar impactos adversos. A ocorrência mais frequente é o *overfitting*, situação na qual o modelo se adapta de forma excessiva aos dados de treinamento, absorvendo até mesmo ruídos e padrões que não são significativos. Isso gera um rendimento reduzido ao

avaliar o modelo com dados inéditos, uma vez que ele compromete a habilidade de generalização. Ainda, a realização de treinamentos por um período prolongado eleva tanto os custos computacionais quanto o tempo requerido para finalizar o processo de treinamento, o que pode representar um obstáculo em dispositivos com recursos limitados, como o *Google Colab*. Finalmente, após um determinado número de *epochs*, os incrementos no desempenho costumam se tornar estáveis. Isso implica que as melhorias tornam-se insignificantes, resultando em períodos adicionais pouco eficazes e, em determinadas situações, até supérfluos [Araújo et al. 2022].

A identificação de objetos após o treinamento revela aspectos de interesse que podem suscitar debates acerca da eficácia da rede em diferentes cenários de teste. Os falsos positivos e os falsos negativos são conceitos essenciais na área de avaliação de modelos de aprendizado de máquina, particularmente em atividades de classificação e identificação de objetos. Os autores relatam categorias específicas de erros que um modelo pode cometer ao tentar classificar de maneira correta ou incorreta os tipos de objetos [Jocher, Chaurasia e Qiu 2023].

Os falsos positivos manifestam-se quando o modelo erroneamente classifica uma amostra negativa (ou seja, um exemplo que não faz parte da classe de interesse) como positiva. De outra forma, o modelo 'reconhece' um objeto que não está presente. Por exemplo, em um modelo de detecção de objetos, um falso positivo pode ocorrer quando identifica erroneamente a presença de um objeto em uma região onde não existe nada. Por outro lado, o falso negativo acontece quando o modelo não consegue reconhecer um exemplo positivo (um exemplo que pertence à classe de interesse). Isto é, o modelo não reconhece um objeto quando este efetivamente se encontra presente. No âmbito de um modelo de identificação de objetos, um falso negativo ocorre quando um objeto perceptível não é reconhecido [Araújo et al. 2022].

Esses dois tipos de falhas têm influência direta na exatidão do modelo. A precisão de um modelo de detecção de objetos representa uma métrica que quantifica a proporção de previsões corretas em comparação ao total de previsões realizadas; assim, essa métrica avalia a aptidão do modelo em reconhecer de maneira adequada as classes positivas.

A precisão (ou *accuracy*) é calculada usando a seguinte fórmula:

$$Precisão = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.1}$$

Onde:

- Verdadeiros Positivos (TP): São os exemplos positivos corretamente identificados pelo modelo.
- Verdadeiros Negativos (TN): São os exemplos negativos corretamente identificados pelo modelo.
- Falsos Positivos (FP): São os exemplos negativos incorretamente classificados como positivos.
- Falsos Negativos (FN): São os exemplos positivos incorretamente classificados como negativos.

Apesar de a precisão constituir uma métrica valiosa, ela pode, por outro lado, induzir a erros, especialmente em situações em que os dados apresentam desequilíbrio (como ocorre quando a classe positiva representa uma fração reduzida do conjunto de dados). Quando um modelo apresenta um elevado número de falsos negativos ou falsos positivos isso pode sinalizar a ocorrência de erros substanciais, o que compromete a eficácia da rede.

Outras métricas são frequentemente usadas para avaliar modelos, como:

Precisão (*Precision*): Mede a proporção de verdadeiros positivos entre todas as previsões positivas feitas pelo modelo. A fórmula é:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Revocação ou Sensibilidade (*Recall*): Mede a proporção de verdadeiros positivos entre todas as instâncias positivas reais. A fórmula é:

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

F1-Score: A média harmônica entre precisão e *recall*, útil para balancear as duas métricas:

$$F1Score = 2 * \frac{Precisão * Recall}{Precisão + Recall} \tag{2.4}$$

A existência de falsos positivos e falsos negativos pode impactar de maneira direta a acurácia de um modelo. Se um modelo gerar um número excessivo de falsos positivos ele pode erroneamente apontar a existência de um objeto, o que pode comprometer a confiança nas previsões realizadas por esse modelo. Por outro lado, se o modelo produzir um elevado número de falsos negativos, poderá não identificar objetos relevantes, o que comprometerá sua eficácia global [Araújo et al. 2022].

Os dois tipos de erro prejudicam o desempenho do modelo e precisam ser reduzidos para assegurar uma precisão adequada e um desempenho eficiente em atividades como a detecção de objetos, o diagnóstico médico ou os sistemas de recomendação. Em contextos nos quais um tipo de erro é considerado mais severo do que outro, como no caso de falsos negativos em diagnósticos de câncer, métricas como *recall* ou F1-score podem ser mais pertinentes para uma avaliação abrangente do modelo [Gomes 2022].

Uma outra métrica que impacta e pode aumentar a precisão das identificações, reduzindo de maneira eficaz a quantidade de falsos positivos e negativos, é o limite de confiança ou *threshold* [Jocher, Chaurasia e Qiu 2023].

A modificação do limite de confiança constitui uma técnica fundamental para aprimorar a eficácia de modelos voltados à detecção de objetos. Esse limite estabelece a probabilidade mínima que um modelo deve apresentar para que uma detecção seja considerada válida. Isto é, trata-se de um parâmetro que estabelece o nível de confiança requerido para que o modelo considere uma previsão como correta, influenciando de maneira direta o número de falsos positivos e falsos negativos. Esse valor atua como um critério de confiabilidade do modelo [Araújo et al. 2022].

Em modelos como o YOLO, cada previsão de objeto é vinculada a uma pontuação de confiança que reflete a probabilidade de uma caixa delimitadora conter um objeto de uma determinada classe. Por exemplo, uma identificação pode

apresentar um grau de confiabilidade de 0,8 quanto à presença de um 'cachorro'. Isso indica que o modelo considera haver 80% de probabilidade de que a referida caixa contenha, de fato, um cachorro. O limite de confiança refere-se ao valor mínimo dessa probabilidade que deve ser alcançado para que a previsão seja considerada válida [Oliveira 2024].

Modificar o limite de confiança, ou ajustar o *treshold do modelo*, exerce uma influência direta sobre a quantidade de falsos positivos e falsos negativos apresentados pelo modelo [Jocher, Chaurasia e Qiu 2023].

- Elevação do Limite de Confiança: Quando o limite de confiança é incrementado, o modelo realiza predições apenas quando possui maior certeza de que o objeto está realmente presente. Por exemplo, ao ajustar o limite para 0,9, o modelo somente considerará um objeto como detectado se a sua confiança for de, no mínimo, 90%. Essa situação, em regra, culmina em uma diminuição dos falsos positivos; isto é, o modelo passará a evitar previsões incorretas em locais desprovidos de objetos. Entretanto, essa abordagem também pode ocasionar um aumento na quantidade de falsos negativos, uma vez que o modelo pode não reconhecer objetos com um nível de confiança um pouco inferior, mas ainda assim significativo [Araújo et al. 2022].
- Diminuição do Limite de Confiança: Por outro lado, caso o limite estabelecido seja excessivamente baixo, o modelo se tornará mais tolerante e passará a reconhecer objetos mesmo quando a probabilidade for inferior, o que pode acarretar um aumento de falsos positivos. Neste cenário, o modelo é capaz de identificar objetos que, na realidade, não estão presentes, embora com uma probabilidade de certeza reduzida. Esse ajuste pode ser benéfico quando a principal meta é identificar o maior número possível de objetos, ainda que isso implique um aumento nas detecções errôneas [Oliveira 2024].

2.6.2 Banco de Dados - *Roboflow*

A plataforma *Roboflow* é extensa e amplamente reconhecida pela criação, pelo gerenciamento e pela preparação de conjuntos de dados voltados ao treinamento de modelos de visão computacional, como os que utilizam a arquitetura YOLO.

Ela oferece uma interface acessível, além de ferramentas avançadas que tornam o processo de interação com dados visuais mais eficaz e adaptável, atendendo tanto a iniciantes quanto a profissionais com experiência. Desde o envio das imagens até a implementação de métodos de pré-processamento e a realização de anotações, o *Roboflow* possibilita a geração de conjuntos de dados altamente adaptados para satisfazer as exigências específicas de cada empreendimento [Oliveira 2024]. Dessa forma, foi feito um banco de dados com 8500 imagens que contém os objetos pertinentes ao presente estudo [BrunoWorkspace 2025]

Um dos aspectos fundamentais na capacitação de modelos para a detecção de objetos é a anotação precisa das imagens. No *Roboflow* os usuários têm a possibilidade de realizar esse procedimento de várias maneiras, conforme a tarefa em pauta. As anotações normalmente envolvem a elaboração de caixas delimitadoras em torno dos objetos de interesse, como pode ser observado nas Figuras 7 e 8, que mostram um exemplo de uma anotação feita em uma imagem do banco de dados, vinculando cada uma delas à respectiva classe pertinente. Essa metodologia possibilita que o modelo assimile a identificação tanto da posição quanto da classificação dos elementos visualizados nas imagens. Em atividades mais intricadas, tais como a segmentação, as marcações são realizadas por meio de máscaras que estabelecem, *pixel* a *pixel*, os contornos precisos dos objetos. Nas classificações de imagens, o propósito consiste em atribuir um rótulo a cada imagem como um todo, sem a exigência de demarcar áreas específicas [Gomes 2022].

Figura 7 – Imagem sem anotação.

Fonte: Retirado de [BrunoWorkspace 2025].

Figura 8 – Imagem com anotação.

Fonte: Retirado de [BrunoWorkspace 2025].

Para além das anotações, o *Roboflow* se sobressai por sua aptidão em efetuar *data augmentation*, uma metodologia essencial para aperfeiçoar a generalização

de modelos de *deep learning*. Essa metodologia consiste em gerar modificações artificiais das imagens originais, ampliando a variedade do conjunto de dados. Por exemplo, é viável modificar o brilho, o contraste e a saturação das imagens, o que auxilia o modelo a se adaptar a condições de iluminação diversas. A incorporação de efeitos como granulação ou ruído possibilita a simulação de situações em que as imagens obtidas possam apresentar interferências ou baixa qualidade, características frequentemente observadas em sistemas do mundo real. Adicionalmente, é viável efetuar rotações, recortes e espelhamentos das imagens, o que permite ampliar a representatividade do conjunto de dados, assegurando que o modelo consiga reconhecer objetos, independentemente de suas orientações ou perspectivas [Oliveira 2024].

A combinação do *Roboflow* com plataformas externas, como *Google Colab* ou *Kaggle*, otimiza o processo de trabalho para o treinamento de redes neurais. Após a realização da preparação e a anotação do conjunto de dados, é possível exportá-lo em diversos formatos, como YOLO, COCO e Pascal VOC, os quais são compatíveis com os *frameworks* destinados ao treinamento. O sistema efetua a organização automática dos dados em pastas destinadas ao treino, à validação e ao teste, assegurando que o usuário não necessite realizar essa separação de forma manual. Ademais, o *Roboflow* cria links diretos para o download dos conjuntos de dados, que são passíveis de utilização nos notebooks de treinamento, dispensando a necessidade de transferir os arquivos manualmente [Jocher, Chaurasia e Qiu 2023].

Ao desenvolver modelos para a identificação de objetos, a qualidade do conjunto de dados gerado no *Roboflow* exerce uma grande influência no desempenho do modelo resultante. A utilização de *data augmentation* desempenha um papel crucial na diminuição da probabilidade de *overfitting*. De uma perspectiva alternativa, metodologias que envolvem o ajuste meticuloso das anotações asseguram que o modelo assimile de forma adequada os padrões visuais dos objetos relevantes, reduzindo erros como falsos positivos e falsos negativos [Oliveira 2024]. Das 8500 imagens que estão na base de dados utilizada, aproximadamente 2000 são de imagens que foram alteradas para melhorar a capacidade do YOLO em fazer identificações mais corretas.

Um outro benefício relevante da integração do *Roboflow* com plataformas

externas é a capacidade de treinar modelos de maneira eficaz, utilizando infraestrutura de alta performance, como GPUs ou TPUs. Esse procedimento agiliza a aprendizagem e possibilita a execução de experimentos com variadas configurações de hiperparâmetros, tais como a quantidade de *epochs*, a taxa de aprendizado e o tamanho do lote. Tais experimentos são fundamentais para adaptar o modelo às particularidades do conjunto de dados, assegurando resultados superiores [Gomes 2019].

A elaboração de bancos de dados consistentes, aliada a metodologias sofisticadas de anotação e pré-processamento, não apenas potencializa a eficácia do treinamento, mas, também, eleva a precisão e a solidez dos modelos implementados [Jocher, Chaurasia e Qiu 2023].

2.6.3 Técnicas para ajuste do limite de confiança

O ajuste do limite de confiança, *treshold*, é essencial para equilibrar a sensibilidade do modelo. Em sistemas de segurança, é desejável um limite de confiança maior para evitar alarmes falsos. Em sistemas de assistência a veículos autônomos, a detecção de objetos é crucial, podendo exigir um limite de confiança mais baixo para evitar negligências, mesmo com mais falsos positivos [Jocher, Chaurasia e Qiu 2023, Araújo et al. 2022].

- Ajuste Manual: O ajuste do limite de confiança (threshold) pode ser feito manualmente, testando diferentes valores de limiar e avaliando o desempenho do modelo com base em métricas como precisão, *recall* e F1-s*core*. Ao monitorar o impacto no desempenho é possível encontrar o valor ideal para o limite de confiança que minimize tanto os falsos positivos quanto os falsos negativos.

  Em aplicações reais, a escolha do *threshold* ideal depende dos objetivos:

- Análise de Curvas de Precisão-*Recall* (PR): Outra abordagem para determinar o limite de confiança é analisar as curvas de precisão-*recall*. Essas curvas mostram o *trade-off* entre precisão e *recall* para diferentes valores de limiar. Ao ajustar o limite de confiança e observar como as métricas se comportam,

pode-se escolher o ponto que oferece o melhor equilíbrio entre as duas métricas, levando em consideração a aplicação específica.

- Uso de *Non-Maximum Suppression* (NMS): O *Non-Maximum Suppression* (NMS) é uma técnica utilizada em conjunto com o ajuste do limite de confiança. Depois que o modelo gera várias caixas de detecção, a NMS é usada para eliminar as previsões redundantes, mantendo apenas a caixa com maior confiança em casos de sobreposição. Isso reduz a quantidade de falsos positivos sem comprometer a detecção dos objetos mais confiáveis.

- Alta *precision* é desejável quando o erro de uma detecção incorreta tem impacto elevado, como em sistemas de alarme automático, onde falsos positivos podem comprometer a credibilidade do sistema.

- Maior *recall* é preferível em contextos em que deixar de detectar um objeto pode trazer riscos maiores, como na vigilância pública ou no monitoramento de atividades suspeitas, onde a omissão é mais grave que um alarme falso.

2.7 Lógica *Fuzzy*

A lógica difusa, conhecida também como lógica *fuzzy*, constitui um domínio da matemática que se ocupa da incerteza e da imprecisão. Em contraste com a lógica clássica, que classifica as proposições como verdadeiras ou falsas, a lógica difusa possibilita que as variáveis adotem valores intermediários, refletindo a incerteza e a gradação das circunstâncias. A proposta foi apresentada por Lotfi Zadeh em 1965 e tem sido amplamente aplicada em sistemas de controle, inteligência artificial, reconhecimento de padrões e, mais recentemente, em visão computacional. A utilização da lógica difusa revela-se especialmente benéfica em circunstâncias nas quais as decisões devem ser fundamentadas em dados incompletos ou imprecisos, o que é uma característica frequente em diversos sistemas da realidade [Morales-Luna 2002].

Na lógica clássica, as proposições são classificadas como verdadeiras ou falsas, o que implica que elas integram o conjunto 0, 1, sendo que 0 simboliza ‘falso’ e 1 simboliza ‘verdadeiro’. Essa modalidade de raciocínio é denominada lógica binária

ou booleana. Entretanto, a realidade, em diversas ocasiões, não se apresenta de forma binária. Em diversos casos, as circunstâncias podem ser 'em parte verdadeiras' ou 'em parte falsas'. Isso se torna particularmente significativo ao abordarmos conceitos como 'quente', 'frio', 'alto', 'baixo', 'rápido' ou 'lento', uma vez que a definição precisa desses termos pode ser influenciada pelo contexto [Morales-Luna 2002].

Essa classe de aplicação amplia essa perspectiva, possibilitando que as proposições apresentem valores intermediários entre 0 e 1. Esses valores são expressos por meio de funções de pertinência, as quais sinalizam o nível de pertencimento de um valor a um conjunto difuso. Por exemplo, em vez de afirmar que 'a temperatura é quente' (categoricamente verdadeiro ou falso), é possível expressar que a temperatura apresenta um grau de 0,7 de ser considerada quente, 0,3 de ser classificada como morna e 0 de ser identificada como fria. Dessa forma, a lógica difusa propicia uma representação mais adaptável e intuitiva de conceitos indeterminados [Eduardo, Vito et al. 2006].

As funções de pertinência constituem um elemento essencial da lógica difusa. Elas estabelecem a maneira pela qual os valores de uma variável são relacionados a um grau de pertinência em um conjunto difuso. Essa função pode se apresentar na forma de uma curva ou como uma função matemática que associa a cada valor de entrada um grau variável entre 0 e 1 [Morales-Luna 2002].

Existem diferentes tipos de funções de pertinência, como:

- Triangular: A função de membros mais simples, com um ponto de pico no valor central.
- Trapezoidal: Similar à triangular, mas com uma base reta, permitindo maior flexibilidade.
- Gaussiana: Uma função de membros mais suave, ideal para representar incertezas contínuas.
- Z e S: Funções com formas em Z ou S que são usadas em sistemas de controle para representar mudanças abruptas em uma variável.

Essas funções auxiliam no gerenciamento da imprecisão, possibilitando que variáveis como 'alta temperatura', 'curta distância' ou 'rápida velocidade' sejam expressas de maneira mais gradual e flexível, em comparação com o que seria viável na lógica convencional [Eduardo, Vito et al. 2006].

De modo semelhante à lógica booleana, na lógica difusa também é possível realizar operações como *AND*, *OR* e *NOT*; entretanto, isso ocorre de forma distinta, haja vista que são apresentados valores contínuos em vez de restringir-se apenas a 0 ou 1.

- *AND* (Conjunção): Em lógica booleana, a operação *AND* retorna verdadeiro apenas quando ambas as proposições são verdadeiras. Na lógica difusa, a operação *AND* retorna o valor mínimo entre os graus de pertencimento das duas variáveis.
- *OR* (Disjunção): Na lógica booleana, *OR* retorna verdadeiro quando pelo menos uma das proposições é verdadeira. Na lógica difusa, *OR* retorna o valor máximo entre os graus de pertencimento.
- (Negação): Na lógica booleana, *NOT* inverte o valor da proposição. Na lógica difusa, o *NOT* é realizado subtraindo o grau de pertencimento do valor 1 [Eduardo, Vito et al. 2006].

Essas operações possibilitam um processamento de informações mais dinâmico e flexível, especialmente em sistemas que necessitam de tomada de decisões em contextos de incerteza [Morales-Luna 2002].

Essa abordagem matemática tem sido utilizada em várias áreas, como controle de sistemas, incluindo regulação de temperatura e velocidade, diagnóstico médico, mecanismos de recomendação, inteligência artificial e processamento de imagens. A sua adaptabilidade em enfrentar incertezas e indefinições a torna uma ferramenta altamente eficaz, principalmente ao interagir com dados ou sistemas que não se apresentam de forma totalmente clara ou precisa. Dessa forma, a lógica *fuzzy* pode ser usada com eficácia para monitoramento como um complemento a modelos de inteligência artificial e processos de treinamento demorados [Muhammad et al.

2021]. Neste artigo, é destacada a importância da lógica *fuzzy* na área de vigilância, direcionando a atenção dos pesquisadores para sua aplicabilidade.

Um dos desafios mais frequentes na área de visão computacional é a segmentação de imagens, isto é, a atividade de dividir uma imagem em distintas regiões com base em atributos como cor, textura ou intensidade. A segmentação convencional emprega limites fixos para diferenciar distintas áreas da imagem; entretanto, essa abordagem pode ser inadequada em fotos com iluminação inconsistente ou com ruídos [Brage e Cañellas 2006].

Nesse contexto, a lógica difusa pode ser aplicada para desenvolver segmentações mais suaves e adaptáveis. Ao invés de categorizar uma região como inteiramente pertencente ou não a um determinado conjunto (como um objeto ou o fundo), a lógica difusa confere a cada *pixel* um grau de pertencimento. Isso implica que a segmentação pode ocorrer de maneira mais gradual e apresenta menor sensibilidade a pequenas alterações na imagem. Ademais, metodologias de *fuzzy clustering*, como o *fuzzy C-Means*, têm sido utilizadas para agrupar *pixels* em agrupamentos difusos, oferecendo uma segmentação mais sólida em imagens que apresentam variações de iluminação ou ruído [Brage e Cañellas 2006].

A identificação de bordas constitui um componente fundamental da visão computacional, sendo crucial para o reconhecimento de contornos de objetos e suas propriedades. Métodos clássicos para a detecção de bordas, tais como o filtro de *Sobel* e o operador de *Canny*, podem apresentar suscetibilidade a ruídos e flutuações de intensidade nas imagens. A lógica difusa pode ser aplicada para otimizar a detecção de bordas, proporcionando maior robustez diante de incertezas e flutuações nos dados [Morales-Luna 2002].

A lógica difusa possibilita uma representação gradual dos contornos, favorecendo uma identificação mais acurada, mesmo em circunstâncias desfavoráveis. Ao invés de reconhecer a borda como uma linha definida, a lógica difusa possibilita a representação da borda como uma transição gradual, o que se revela benéfico ao lidar com imagens que apresentam bordas suaves ou esbatidas [Brage e Cañellas 2006].

Os Sistemas de Inferência Difusa encontram vasta aplicação em várias disciplinas, abrangendo, entre outros, a visão computacional. Trata-se de um

conjunto de normas e de uma função de agregação que converte as entradas difusas em uma saída difusa. Na área de visão computacional, sistemas baseados em Lógica *Fuzzy* (FIS) podem ser empregados para a tomada de decisões relacionadas à classificação de objetos, como verificar se um determinado objeto identificado em uma imagem corresponde realmente ao que o sistema busca, levando em conta as incertezas associadas às propriedades da imagem [Eduardo, Vito et al. 2006].

Em um sistema destinado à detecção de objetos, por exemplo, um Sistema de Inferência *Fuzzy* (FIS) pode analisar atributos como cor, textura e forma de um objeto, atribuindo diferentes níveis de pertencimento a diversas classes de objetos. Em lugar de determinar de forma estrita se o objeto se enquadra em uma classe específica, o sistema é capaz de apresentar uma resposta probabilística, sinalizando o nível de confiança em relação à possibilidade de o objeto pertencer a uma categoria particular [Eduardo, Vito et al. 2006].

A lógica difusa proporciona uma abordagem eficaz para gerenciar incertezas e imprecisões em sistemas de visão computacional. Ela possibilita que as máquinas realizem decisões e interpretações mais 'humanas', fundamentadas em informações que podem ser imprecisas ou vagamente definidas. A adoção de lógica difusa em atividades de segmentação, identificação de bordas e sistemas de inferência difusa pode elevar de maneira significativa o desempenho de sistemas de visão computacional, conferindo-lhes maior robustez e a capacidade de enfrentar as complexidades do mundo real [Brage e Cañellas 2006].

A combinação de técnicas de lógica difusa com recursos como o OpenCV proporciona a pesquisadores e desenvolvedores uma ferramenta robusta para desenvolver sistemas de visão computacional que sejam mais eficazes e adaptáveis. Isso cria novas oportunidades para o progresso tecnológico em campos como automação, robótica, segurança, medicina, entre outros [Mosselman, Weenink e Lindegaard 2018].

2.8 Conclusão

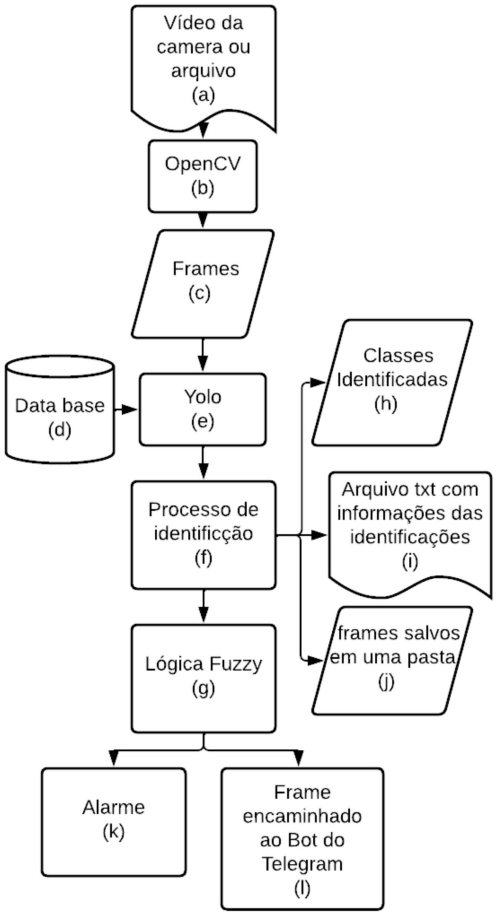
Em síntese, a Lógica *Fuzzy* demonstra ser uma ferramenta eficiente para lidar com incertezas e imprecisões em análise de dados. Sua aplicação permite uma

interpretação mais flexível e realista, superando limitações de abordagens rígidas e possibilitando uma avaliação mais abrangente das situações, especialmente em cenários críticos, como a detecção de objetos que podem apresentar algum grau de periculosidade. Neste trabalho, a utilização da Lógica *Fuzzy* foi implementada em conjunto com o modelo YOLO, uma vez que essa integração apresentou resultados promissores. A combinação das duas abordagens mostra-se eficaz ao unir a rapidez e acurácia do YOLO na identificação de objetos com a capacidade da lógica difusa de interpretar variáveis contínuas e incertas, como distância, velocidade e nível de risco. O sistema implementado utiliza o ROS para gerenciar cada módulo, incluindo a identificação de objetos, a coleta de dados e a avaliação adaptativa de cenários, possibilitando a geração de alertas mais próximos da realidade. Dessa forma, a integração entre YOLO e Lógica *Fuzzy* contribui de maneira significativa para aumentar a robustez, a precisão e a confiabilidade do sistema de monitoramento proposto, configurando-se como uma solução eficiente e promissora para aplicações em ambientes dinâmicos e de alto risco.

3 Metodologia

A Figura 9 apresenta o fluxograma geral da metodologia adotada. Inicialmente, o arquivo da fonte, seja um vídeo de teste ou um vídeo da câmera, foi transformado em uma imagem no formato compatível com o *OpenCV*. Os *frames*, que são, essencialmente, imagens isoladas, ao serem integrados geram a ilusão de movimento; assim, a análise do vídeo foi conduzida através de imagens individuais correspondentes a cada segmento da gravação.

Figura 9 – Fluxograma do projeto.



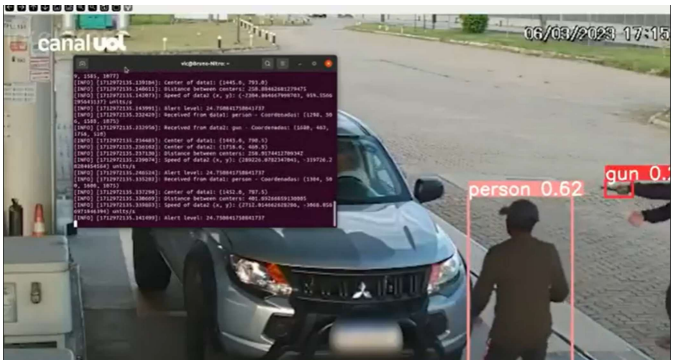
Fonte: Elaborado pelo autor.

3.1 Vídeos de teste e câmera.

Para testar a rede neural foram selecionados cinco vídeos distintos, retirados do *YouTube*, nos quais foram avaliadas situações em que as duas classes, *person* e *gun*, foram observadas, conforme indicado na Figura 9 (a). Cada vídeo apresentou condições que tornavam a identificação da classe *gun* mais complicada pelo YOLO, uma vez que armas são, em sua maioria, objetos pequenos e mais suscetíveis a gerar falsos positivos e negativos nos resultados do procedimento de identificação.

- Vídeo 1: Uma gravação de um assalto a um cliente em um posto de gasolina. Não há muitos objetos, mas há movimentação, objetos com brilho e, principalmente, distância entre a câmera e os objetos (Figura10).

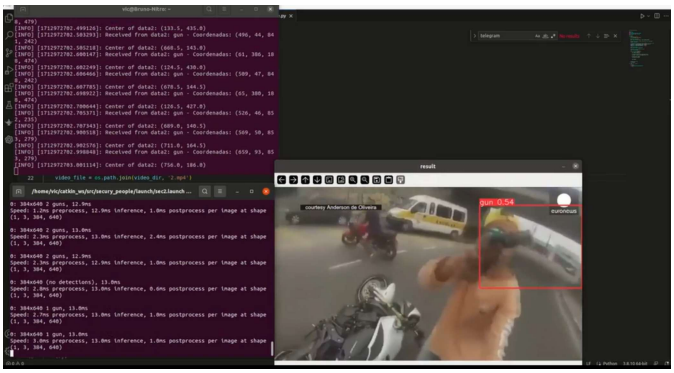
Figura 10 – *Print* do vídeo 1.



Fonte: https://encurtador.com.br/5DGt9

- Vídeo 2: Um assalto a motociclista em uma avenida movimentada; há movimentação rápida de diversos objetos nas imagens (Figura 11).

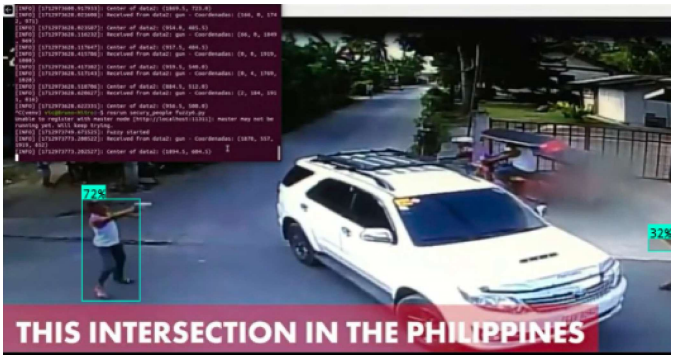
Figura 11 – *Print* do vídeo 2.



Fonte: https://encurtador.com.br/Cnn1f

- Vídeo 3: Vídeo de uma câmera de rua, onde ocorre uma troca de tiros entre pessoas. A maior dificuldade desse vídeo para o YOLO é a distância para a identificação de objetos e a velocidade de movimentação das pessoas (Figura 12).

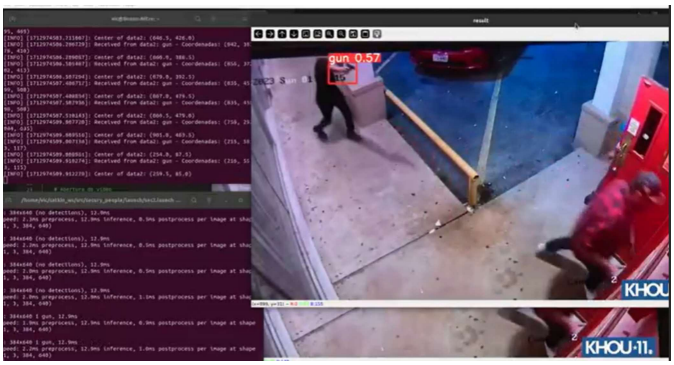
Figura 12 – *Print* do vídeo 3.



Fonte: https://encurtador.com.br/yMTyZ

- Vídeo 4: Vídeo de uma câmera na entrada de uma residência, com diversas pessoas segurando objetos. Há pouca movimentação, mas em um momento uma das pessoas saca uma arma e atira(Figura 13).

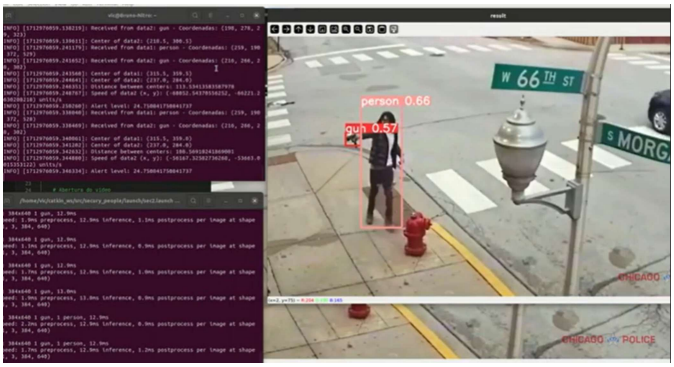
Figura 13 – *Print* do vídeo 4.



Fonte: https://encurtador.com.br/Fk9Si

- Vídeo 5: Esse vídeo apresenta a segurança de uma loja que mostra tanto o ambiente interno quanto o ambiente externo ao estabelecimento. Há diversos objetos que podem ser confundidos com armas (Figura 14).

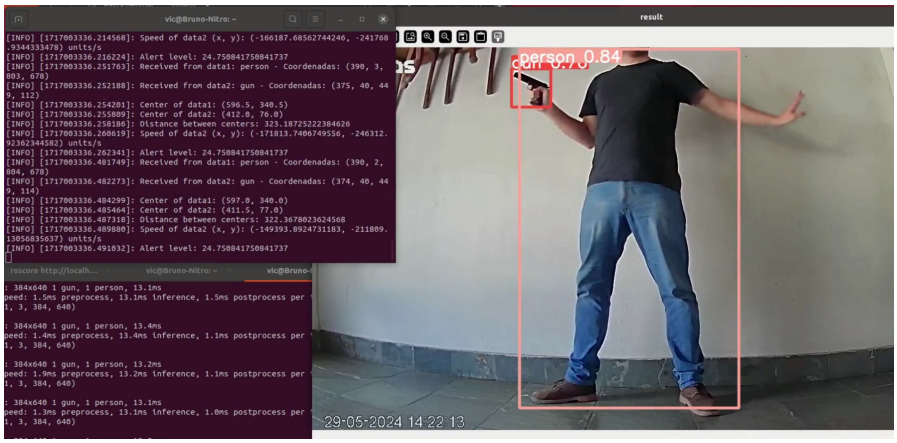
Figura 14 – *Print* do vídeo 5.



Fonte: https://encurtador.com.br/G8hBI

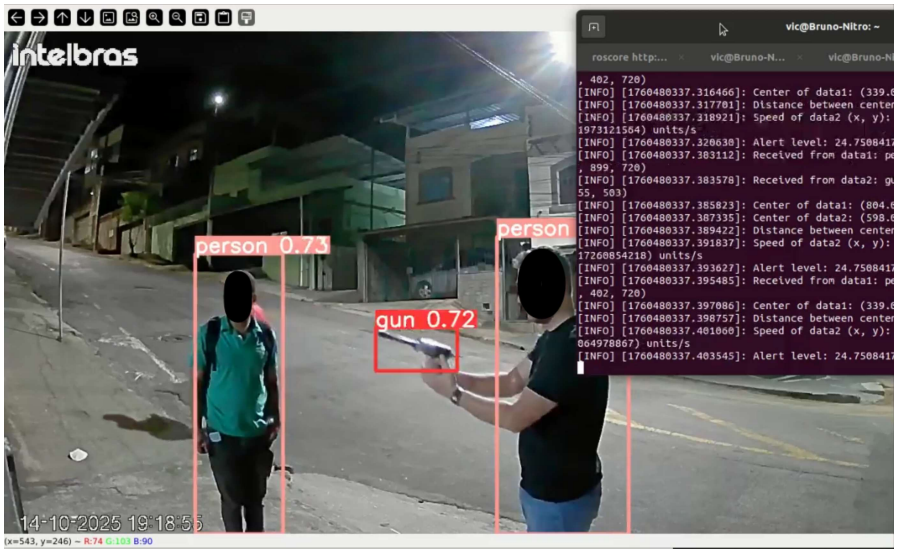
Além desses vídeos, também foi utilizada a Câmera IP da *Intelbras,* que se comunica com o nó do ROS, Video2Image, pelo protocolo de rede *Real time stream protocol* (RTSP), que permite o envio de imagens da câmera para o programa com uma boa velocidade, dependendo apenas da conexão com a internet. Para essa aplicação, foram feitos dois testes, primeiramente em um ambiente interno, exibido na Figura 15 e em um ambiente externo, como na Figura 16.

Figura 15 – Vídeo da câmera aplicada ao ambiente interno.



Fonte: Elaborado pelo autor.

Figura 16 – Vídeo da câmera aplicada ao ambiente externo.



Fonte: Elaborado pelo autor.

A partir disso, o Nó 1, que inicia o programa, é o responsável pela conversão do arquivo de vídeo ou da imagem da câmera em *frames*, utilizando o OpenCV,

apresentado na Figura9 tópico (b) e (c).

3.2 YOLO Treinado

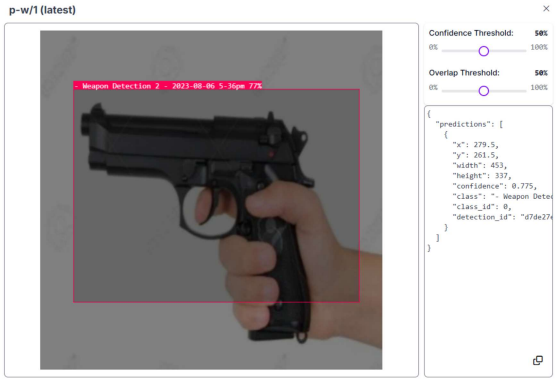
A partir dessa conversão, o *framework* ROS utiliza um método para identificação de objetos, empregando YOLO, que foi previamente treinado com um conjunto de dados, indicado pelo tópico (d) da Figura 9 (composto por imagens que incluem os objetos relevantes para o estudo [BrunoWorkspace 2025]. A partir desse instante, o *framework* dá início a um procedimento de coleta de dados e análises, bem como à determinação das coordenadas das classes e à realização de cálculos de distância e velocidade.

O processo de identificação pelo YOLO, representado na Figura 9 pelo tópico (f), gera três resultados diferentes:

- Um vídeo que mostra os quadros combinados, no qual são visíveis as *bounding boxes* que circundam os objetos que estão em destaque no *frame* (tópico (h) da Figura 9).
- Um arquivo no formato .txt com informações detalhadas a respeito da data e do horário em que as detecções foram realizadas, abrangendo as classes que são pertinentes a este trabalho (tópico (i) da Figura 9).
- Uma pasta na qual estão armazenados todos os *frames* que mostram as classes que foram identificadas durante o processo de análise (tópico (j) da Figura 9).

Além disso, o YOLO publica em dois tópicos separados (data1 e data2) da Figura23, as coordenadas dos objetos nos *frames*, como exibido nas Figuras 17 e 18.

Figura 17 – Informações das coordenadas identificadas nas imagens.



Fonte: Adaptado de [BrunoWorkspace 2025].

Figura 18 – *Log* no arquivo txt.

Arquivo Editar Exibir


```
{
  "predictions": [
    {
      "x": 279.5,
      "y": 261.5,
      "width": 453,
      "height": 337,
      "confidence": 0.775,
      "class": "- Weapon Detection 2 - 2023-08-06 5-36pm",
      "class_id": 0,
      "detection_id": "04446905-852d-4eef-a9ea-5378dd89f4f9"
    }
  ]
}
```


Fonte: Elaborado pelo autor.

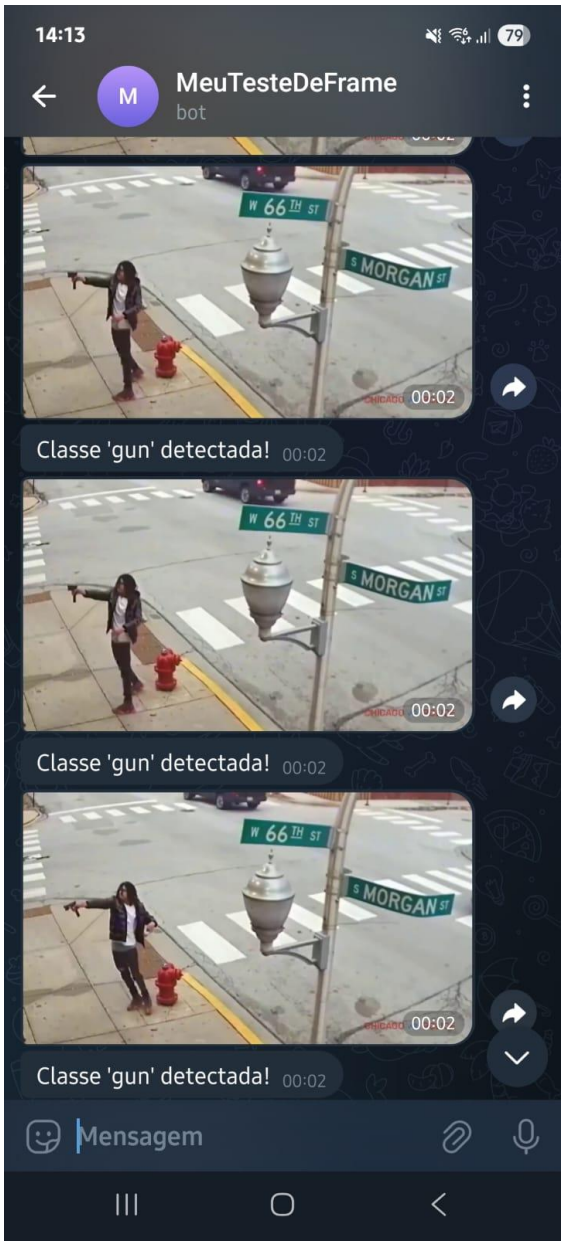
Cada célula da grade prevê as caixas delimitadoras, assim como as probabilidades de classe. A predição da caixa delimitadora possui cinco elementos: (x, y, w, h, confiança). As coordenadas (x, y) representam o centro da caixa, em relação à localização da célula da grade (vale lembrar que, se o centro da caixa não estiver localizado dentro da célula da grade, a responsabilidade por ela não recai sobre essa célula). Os elementos w e h representam, respectivamente, a largura e a altura do objeto em relação ao tamanho da imagem. Também são exibidas as classes e a confiança da detecção. As informações de coordenadas das classes são detectadas em tópicos separados e processadas pelo algoritmo *fuzzy*, o qual realiza análises de periculosidade e estabelece níveis de alerta.

O banco de dados da Figura 9 (tópico (d)) , utilizado no treinamento da rede, foi elaborado na plataforma *Roboflow*. Primeiramente, foram selecionadas imagens que atendiam às especificações dos objetos que deveriam ser identificados: pessoas e armas de fogo. Após o treinamento, pôde-se observar que as versões melhoraram seus resultados ao mesmo tempo em que passaram a demandar mais processamento do hardware o que, consequentemente, elevou o tempo necessário para a realização da operação.

3.3 Nó *Fuzzy*

O nó *Fuzzy*, representado pelo tópico (g) da Figura 9, reúne os dados de coordenadas das classes e faz os cálculos de distância e velocidade entre os objetos. Dessa forma, ao analisar essas informações juntamente com as funções de pertinência, são gerados os índices de periculosidade e alerta. Nesse nó é realizada a tomada de decisão sobre a mensagem ser encaminhada ao *bot* do Telegram ou não, dependendo do nível de alerta estipulado. Esse processo é representado pelos tópicos (k) e (l) da Figura 9. O tópico (l) também é apresentado na Figura19, que apresenta o *bot* recebendo os *frames* enviados.

Figura 19 – Print do bot do Telegram recebendo o *frame*.



Fonte: Elaborado pelo autor.

3.4 Aplicação da Lógica Fuzzy

Para o modelo *fuzzy* desenvolvido foram definidas duas variáveis de entrada:

- A primeira é a distância entre os centros das caixas delimitadoras de objetos *person* e *gun*, calculada por distância entre dois pontos (Equação 3.1) utilizando as coordenadas coletadas pelo algoritmo de detecção [Brage e Cañellas 2006].

$$D^2 = (X1 - X2)^2 + (Y1 - Y2)^2 \tag{3.1}$$

- Em que:
  - (X1, Y1) são as coordenadas do centro da *box* do objeto '*gun*';
  - (X2, Y2) são as coordenadas do centro da *box* do objeto '*person*';
  - D é a distância entre os objetos.
- A segunda variável é a velocidade de movimentação desses objetos que, de acordo com [Mosselman, Weenink e Lindegaard 2018], aponta dados como a movimentação periculosa em atuações criminosas, determinada pela Equação 3.2 da distância em pixels de movimentação da classe em um intervalo de tempo.

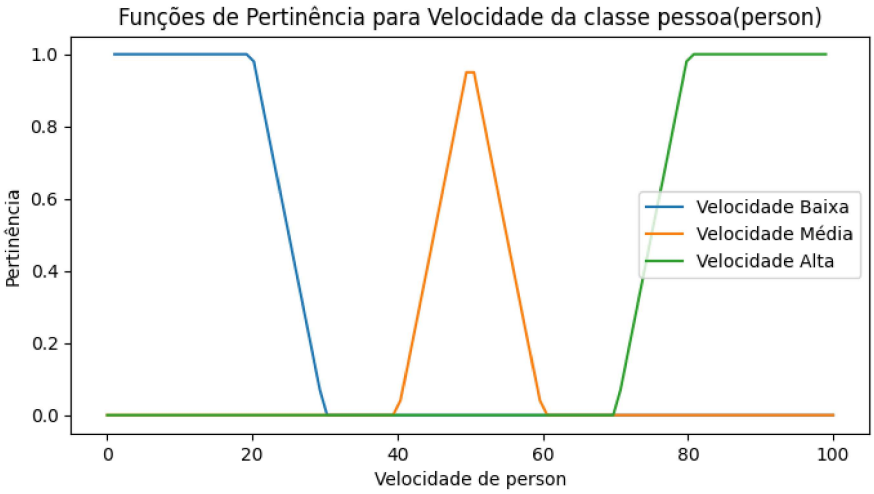
$$Velocidade = \frac{df distancia}{dtempo} \tag{3.2}$$

A partir desse ponto, foram determinadas normas do que torna uma situação perigosa ou não. A análise adotou os seguintes parâmetros: todas as situações de alerta são vinculadas à presença simultânea de pessoas e armas juntas, então, se não houver um ou outro, o risco cai. Caso a distância entre pessoas e armas seja baixa e a velocidade de armas seja baixa, o alerta é alto, o que significa que a arma está sendo apontada para alguém ou disparada. Caso a distância seja baixa e a velocidade da arma seja alta, indica uma situação em que alguém está correndo armado, então é uma situação de alerta para um possível incidente. Já quando a distância é alta e a velocidade da arma baixa, significa que a arma foi largada, então o nível de alerta é baixo [Morales-Luna 2002].

De acordo com as regras apontadas anteriormente, que graficamente são representadas na Figura 20, uma pessoa transitando com velocidade média não

representa perigo, por isso a função de pertinência trapezoidal assume valores de pertinência mínimos. Por outro lado, os extremos de velocidade baixa e alta estão com a pertinência no máximo, pois atendem à situação de alguma abordagem ou fuga [Morales-Luna 2002].

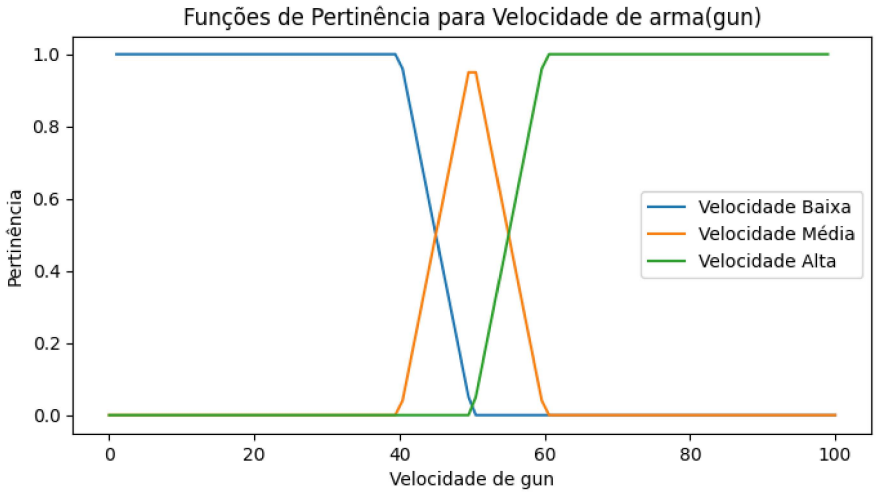
Figura 20 – Função de pertinência para a classe *person*.



Fonte: Elaborado pelo autor.

Da mesma forma, a Figura 21 representa graficamente as regras apresentadas para a utilização de arma. Como apresentado anteriormente, há maiores probabilidades de ocorrência de uma situação de perigo quando a movimentação da pessoa está mais lenta ou mais rápida, daí maiores alertas para velocidades nos dois extremos têm a pertinência máxima. O risco médio é refletido por uma função de pertinência quase triangular que indica situações que podem até mesmo serem confundidas com um falso positivo de que um objeto se move causado por *pixels* mal definidos ou por sobreposição de cores [Brage e Cañellas 2006].

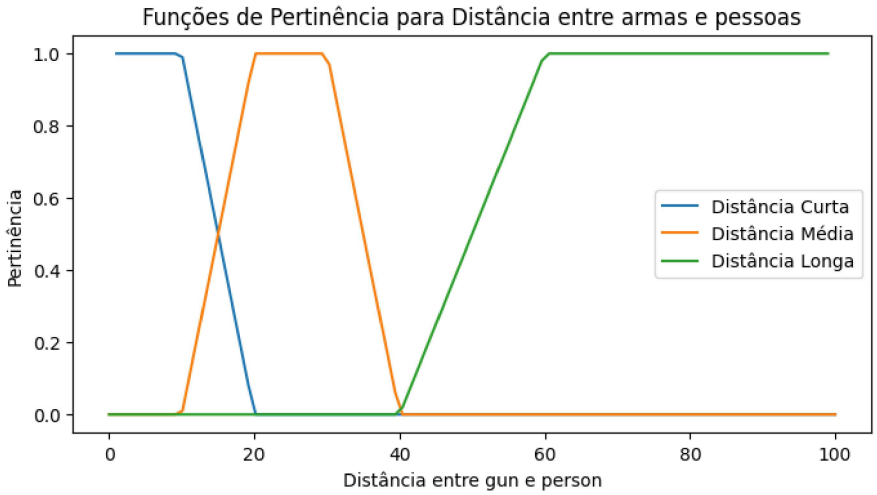
Figura 21 – Função pertinência para a classe *gun*.



Fonte: Elaborado pelo autor.

No caso da Figura 22, a partir das regras apresentadas, não há alerta quando a distância entre pessoas e armas for grande, pois seria uma situação em que ou a arma foi solta ou um falso positivo ocorreu. Para a situação em que a distância é menor, infere-se que pode ser uma circunstância de alerta, quando a pessoa e a arma estiverem em uma disposição de diagonal ou de sobreposição (o que indica o saque da arma ou que esta está sendo segurada próxima ao corpo). No que tange à pertinência de distâncias médias, o nível de perigo é relacionado aos cálculos de distâncias relacionadas a interações menos inclinadas e mais horizontais. Porém, essas medidas sofrem interferência de acordo com a proximidade da câmera, quanto mais próximas, maior é a distância entre os centros de objetos [Brage e Cañellas 2006].

Figura 22 – Função de pertinência entre *person* e *gun*.



Fonte: Elaborado pelo autor.

Os valores numéricos das variáveis, velocidade e distância, foram convertidos em graus de pertinência utilizando funções de pertinência trapezoidais. Assim, a fase de fuzzificação transformou as medidas obtidas dos *frames* — originalmente expressas em unidades discretas — em classificações linguísticas como “baixa”, “média” e “alta”.

Na etapa de inferência, as regras fuzzy foram formuladas para refletir a relação lógica entre as variáveis analisadas. As principais regras estabelecidas foram:

- se a velocidade da arma é baixa ou alta, então há perigo;
- se a velocidade da pessoa é baixa ou alta, então não há perigo;
- se a distância entre pessoa e arma é curta, então há perigo, enquanto distâncias elevadas indicam ausência de risco.

  Tais regras foram aplicadas de forma simultânea, sendo as condições combinadas por operadores lógicos do tipo “E” (mínimo) e “OU” (máximo), de modo a obter a ativação dos conjuntos fuzzy de saída correspondentes ao grau de risco.

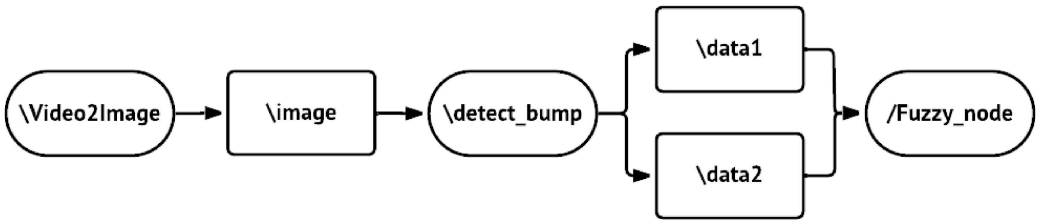
Por fim, a etapa de defuzzificação foi executada pelo método *centroid*. Esse procedimento calcula o centro de gravidade da área resultante das funções de saída, produzindo um valor numérico contínuo que expressa o nível final de alerta. Assim, valores próximos a 0 indicam ausência de perigo, enquanto valores mais elevados correspondem a situações de risco potencial. Essa abordagem possibilitou que o sistema ajustasse dinamicamente o nível de alerta em função da variação das entradas, contribuindo para reduzir alarmes falsos e aumentar a coerência das detecções em tempo real.

.

3.5 Estrutura final

A estrutura do programa desenvolvido ficou de acordo com o *rqt_graph* do ROS (Figura 23).

Figura 23 – Nós e tópicos.



Fonte: Elaborado pelo autor.

O Nó 1, denominado **Video2Image**, é o responsável por iniciar o programa realizando a conversão do arquivo de vídeo ou do fluxo de imagens da câmera em *frames*. Após a conversão, esses *frames* são publicados no tópico **Image**. Em seguida, os *frames* são processados pelo Nó 2, **detect_bump**, que utiliza o YOLOv8 para identificar os objetos presentes nas imagens, extraindo informações relevantes de cada classe, como as coordenadas dos objetos, que são publicadas em dois tópicos distintos: **data1** (*gun*) e **data2** (*people*). Esse nó também gera um arquivo de texto contendo os dados das detecções e armazena os *frames* que contêm as classes de interesse para o estudo.

Por fim, o Nó 3, **fuzzy_node**, aplica as métricas das funções de pertinência da lógica *fuzzy* para analisar os dados recebidos, calculando as distâncias e velocidades entre os objetos, de forma a inferir o nível de periculosidade conforme as métricas definidas. Além disso, esse nó é responsável por enviar o *frame* correspondente ao bot do Telegram e emitir alertas para o sistema, notificando o operador quando necessário.

3.6 Conclusão

Diante do exposto, a metodologia definida para este trabalho integrou de forma eficiente diferentes ferramentas e tecnologias, combinando a captura de imagens por meio de vídeos reais e uma câmera IP, o processamento dos dados com o *framework* ROS, a identificação de objetos pelo YOLOv8 treinado no Roboflow e a aplicação da lógica *fuzzy* para análise de risco. A estrutura modular dos nós desenvolvidos garante a conversão, detecção, análise e comunicação dos resultados de forma automatizada, com geração de alertas para o usuário final via *bot* do Telegram. O uso de hardware compatível e de fácil acesso reforça a viabilidade prática do sistema proposto, possibilitando validar a performance do modelo em cenários reais de monitoramento conforme as necessidades de identificação de pessoas e armas de fogo em diferentes contextos.

4 Resultados e Discussão

Neste estudo, foi desenvolvido um sistema de identificação de objetos, com ênfase na detecção de indivíduos e armas, utilizando visão computacional. A metodologia empregada incorporou o framework ROS para a organização dos componentes, o Roboflow para a construção e anotação do banco de dados, o modelo YOLO para a detecção ágil e exata dos objetos e a lógica fuzzy para aprimorar a filtragem dos resultados, diminuindo os falsos positivos e negativos, e, consequentemente, elevando a confiabilidade do sistema.

Com base nos resultados alcançados, esse capítulo oferece uma análise acerca das principais descobertas, limitações e contribuições do estudo. A implementação da lógica *fuzzy*, combinada com o filtro de confiabilidade na saída do modelo, revelou-se essencial para reduzir alarmes falsos, otimizando a utilização dos recursos computacionais e aprimorando a aplicabilidade prática do sistema em contextos reais.

Para o desenvolvimento deste projeto, foi disposto um Notebook Acer nitro5 com 16gb de memória ram, um Ryzen 7 5800 com placa de vídeo Nvidia 1650. Além disso, foi utilizado um *bot* programável no Telegram, que informa ao usuário qual classe importante à rede neural foi identificada nas imagens da câmera, para fazer a integração entre o sistema e usuário final. A câmera IP (Protocolo de Internet) utilizada foi a Intelbras IMc X1 (Figura 24), por ser um modelo popular e de baixo custo. Suas especificações encontram-se na Tabela 1.

Figura 24 – Intelbras IMc X1.

Fonte: Intelbras

Tabela 1 – Especificações da câmera.

Sensor	1/4” 2MP ProgressiveCMOS
Pixels efetivos	1280 (H) x 720 (V)
Linhas horizontais	1280 (H)
Resolução real	HD (720p)
Lente	2,6 mm
Ângulo de visão	114°(diagonal),93°(H),50°(V)
Zoom digital	8x
Alcance IR	10m
IR inteligente	Sim
Comprimento de onda LD IR	850nm
Sensibilidade	6 lux (IR ligado)
Armazenamento	MicroSD até 256GB
Day & Night	Automático / P&B
Troca Automática do Filtro (ICR)	Sim

Fonte: https://backend.intelbras.com/sites/default/files/2023-01/datasheet-imx1-pt.pdf

4.1 Resultados e discussão do treinamento do YOLO

A partir da metodologia exposta anteriormente, foi determinada a confiabilidade da rede e, utilizando-se os parâmetros de avaliação da *Ultralytics* [Jocher, Chaurasia e Qiu 2023], chegou-se aos valores da melhor *epoch* do treinamento na última versão do banco de dados.

Os dados posteriores ao treino podem ser consultados na Tabela 2.

Tabela 2 – Dados do YOLOv8s

Model	YOLOv8	Modelo treinado
Size(pixels)	640	640
mAPval50-95	44,9	47,999
SpeedCPU ONNX(ms)	128,4	130,523
SpeedA100 TensorRT(ms)	1,2	1,3122
params(M)	11,2	11,2
Flops(B)	28,6	28,4

Fonte: Ultralytics e elaborado pelo autor.

O mAP é a média da métrica Precisão Média em todas as classes de um

modelo e é frequentemente usado para avaliar o desempenho global do modelo em múltiplas classes de objetos. Os FLOPs (*Floating Point Operations Per Second*) são uma métrica que indica a quantidade de operações de ponto flutuante necessárias para processar uma única imagem de entrada através da rede neural. Quanto maior o número de FLOPs, maior a carga computacional e, muitas vezes, maior o tempo de inferência (tempo para realizar previsões) do modelo.

Quantos menores as métricas das funções *loss*, melhor é a capacidade das redes em captar os objetos e localizá-los em relação à caixa delimitadora, bem como suas coordenadas (Tabela 3).

- Box *Loss*: Avalia a precisão das coordenadas das caixas delimitadoras preditas.
- Cls *Loss*: Avalia a precisão da classificação dos objetos dentro das caixas delimitadoras.
- DFL *Loss*: Refinamento adicional da localização das caixas delimitadoras para melhorar a precisão das bordas.

Tabela 3 – Dados do YOLOv8s frente ao treinamento - perdas.

box_loss	cls_loss	dfl_loss
0,43887	0,32424	1.0364

Fonte: Elaborado pelo autor.

Outros dois dados apresentados na Tabela 4, a *Precision* e o *Recall*, possuem conceitos semelhantes, porém com distinções importantes: o primeiro avalia a exatidão das detecções realizadas pela rede, isto é, quantas das predições são realmente corretas; enquanto o segundo mede a capacidade do modelo de identificar todas as instâncias relevantes, mesmo que algumas predições estejam erradas. Assim, observa-se que o treinamento resultou em uma rede com *precision* de 0,89852, o que indica que a maioria das detecções do modelo está correta, evidenciando um bom desempenho na identificação de objetos.

Já o mAP50-95 corresponde à média das precisões para múltiplos limiares de IoU, variando de 0,5 a 0,95, em incrementos de 0,05. Por oferecer uma visão mais

ampla do desempenho do modelo sob diferentes graus de exigência na sobreposição, essa métrica é considerada mais rigorosa e completa. Quanto maior seu valor mais robusto é o modelo, indicando boa performance mesmo sob condições variadas de detecção. Esse foi o critério adotado para selecionar a melhor *epoch* entre as trezentas do processo de treinamento.

Tabela 4 – Dados do YOLOv8s frente ao treinamento - métricas.

precision(B)	*recall*(B)	mAP50(B)	mAP50-95(B)
0,89852	0,69823	81.84	47,999

Fonte: Elaborado pelo autor.

O valor de Precision igual a 0,89852 indica que aproximadamente 89,85% das detecções realizadas pelo modelo foram corretas, ou seja, a grande maioria das predições feitas correspondeu, de fato, a objetos reais presentes na cena. Esse é um resultado muito satisfatório em tarefas de detecção de objetos, especialmente quando se trabalha com imagens de câmeras de vigilância ou monitoramento, onde estão presentes ruídos visuais, oclusões, variações de iluminação e movimento contínuo.

Embora a precisão (*precision*) tenha apresentado um valor elevado, indicando que o modelo comete poucos falsos positivos, o valor de *recall* mostra-se relativamente baixo. Essa métrica representa a capacidade do modelo em encontrar corretamente todos os objetos presentes na cena, ou seja, mede a proporção de verdadeiros positivos em relação ao total de objetos reais. Portanto, esse valor indica que o modelo tem uma alta confiabilidade para identificar corretamente os objetos que detecta, o que representa um desempenho desejável para aplicações em tempo real ou em cenários críticos, como a detecção de comportamentos suspeitos ou eventos anômalos em vídeos de câmeras.

Um valor de *recall* reduzido indica que o modelo não está detectando todos os objetos que deveria, resultando em um número considerável de falsos negativos — situações em que um objeto real, como uma arma ou uma pessoa, não é identificado. Em aplicações práticas, especialmente em contextos de segurança pública, monitoramento urbano ou sistemas de vigilância automatizados, essa limitação tem implicações sérias.

Em outras palavras, há um trade-off entre precisão e *recall*: enquanto a alta precisão evita alertas indevidos, o baixo *recall* limita a capacidade de cobertura do sistema. Em um contexto de segurança, é preferível buscar um equilíbrio entre ambas as métricas, mesmo que isso implique tolerar um número ligeiramente maior de falsos positivos, desde que se reduza a chance de não detectar ameaças reais.

Durante a etapa de validação, é essencial avaliar o desempenho do modelo frente a dados não vistos, isto é, que não participaram do ajuste direto dos pesos da rede. São monitoradas métricas de perda como:

- *val/box_loss:* Mede o erro de regressão das *bounding boxes* (diferença entre a caixa predita e a caixa real). Quanto menor, melhor está o ajuste das caixas aos objetos.
- *val/cls_loss:* Mede o erro de classificação dos objetos detectados. Quanto menor, melhor o modelo está identificando corretamente as classes dos objetos.
- *val/dfl_loss (Distribution Focal Loss):* utilizada para melhorar a localização precisa das caixas (especialmente em modelos como YOLOv8).

Conforme apresentado na Tabela 5, os valores obtidos para as métricas de perda durante a validação foram consideravelmente elevados quando comparados aos parâmetros de referência da *Ultralytics* para um modelo bem ajustado. Esses resultados sugerem que o desempenho do modelo pode ser aprimorado, especialmente em relação à localização das caixas delimitadoras e à classificação correta das classes durante a validação.

As imagens da classe “arma” eram predominantemente compostas por objetos próximos à câmera, o que significa que as armas ocupavam áreas maiores no quadro. Durante o processo de treinamento, o modelo aprendeu com exemplos em que o objeto possuía grande representatividade espacial. Assim, ao ser exposto a imagens em que as armas aparecem distantes, parcialmente visíveis ou menores, o modelo enfrenta maior dificuldade para identificá-las corretamente, resultando em maiores erros de localização (box loss) e classificação (cls loss).

Outro ponto relevante é que, embora o treinamento tenha sido realizado por 300 *epochs*, esse número pode ter sido suficiente para alcançar a convergência

ideal. O aumento gradual da perda durante a validação pode indicar overfitting, ou seja, o modelo se ajustou excessivamente às amostras de treinamento e perdeu capacidade de generalização nos dados de validação.

Tabela 5 – Dados do YOLOv8s frente ao treinamento - validação.

val/box_loss	val/cls_loss	val/dfl_loss
1,963	1,1941	2,7702

Fonte: Elaborado pelo autor.

As medidas lr/pg0, lr/pg1 e lr/pg2, exibidas na Tabela 6, referem-se a taxas de aprendizado para diferentes partes do modelo: pg0 que é relacionado à *backbone*, parte responsável pela captação de características da imagem; pg1 que está ligado às camadas responsáveis pela detecção de objetos, conhecidas como cabeçalhos; e pg2 que abrange parâmetros complementares, como normalizações, que se refere a técnicas que ajustam os valores dos parâmetros para melhorar o desempenho e a estabilidade do treinamento. A influência dessas variáveis no desempenho do modelo está diretamente ligada à taxa com que seus parâmetros são atualizados. Uma taxa de aprendizado excessivamente elevada em pg0 pode resultar na perda de características valiosas que foram adquiridas durante o pré-treinamento pela *backbone*, enquanto uma taxa de aprendizado muito reduzida em pg1 pode dificultar a adaptação eficaz das camadas de detecção ao novo conjunto de dados. Já no caso de pg2, é particularmente sensível a grandes variações e, por isso, costuma receber uma taxa de aprendizado menor.

Tabela 6 – Dados do YOLOv8s frente ao treinamento - taxa de aprendizado.

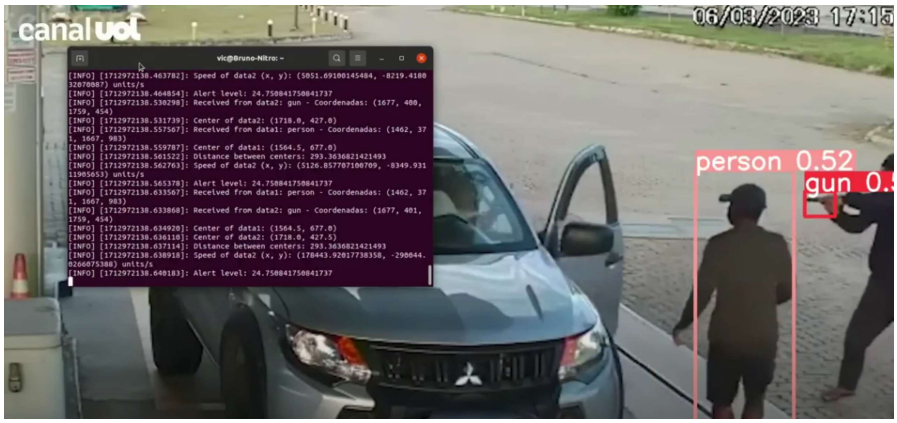
lr/pg0	lr/pg1	lr/pg2
0,003664	0,003664	0,003664

Fonte: Elaborado pelo autor.

Esses valores foram determinados no final do treinamento, na *epoch* com melhores índices de detecção. Mas é notável que esses valores funcionaram bem para o *dataset* no atual formato, em outras condições de treino, mas pode não ser ótimo quando aplicado em outro *dataset* com imagens e pesos diferentes.

Esta rede foi capaz de analisar, por meio da lógica *fuzzy,* os momentos em que um alerta deve ser acionado por uma *flag* no terminal do ROS. Por exemplo, na Figura 25, retirada de um dos vídeos de amostra, é possível observar a identificação de objetos pelo YOLO e o terminal com a rede *fuzzy* avaliando as medidas de periculosidade.

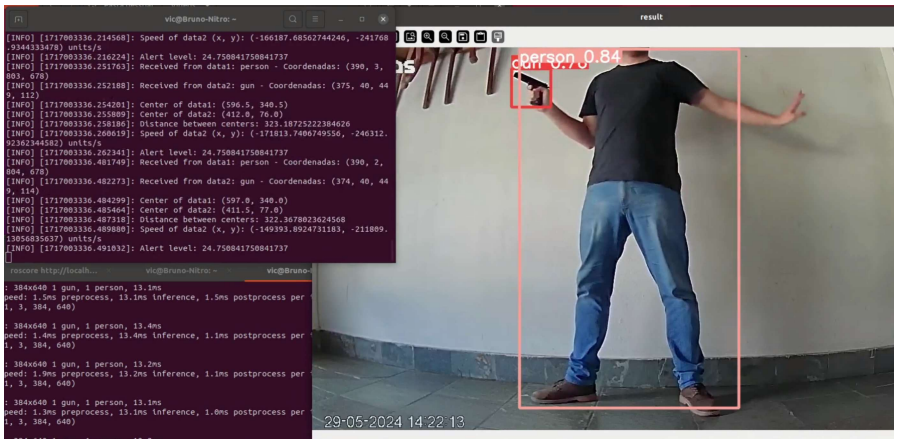
Figura 25 – Captura de tela do vídeo1.



Fonte: Elaborado pelo autor.

O mesmo teste foi realizado usando a imagem da câmera Intelbras com o propósito de exibir a identificação de objetos e o funcionamento da lógica *fuzzy* com a aplicação do modelo nos *frames* advindos da câmera. Para tal, foram utilizados o *Real Time Stream Protocol* (RTSP) e uma pistola de *airsoft*. Os resultados obtidos podem ser visualizados na Figura 26.

Figura 26 – Captura da câmera.



Fonte: Elaborado pelo autor.

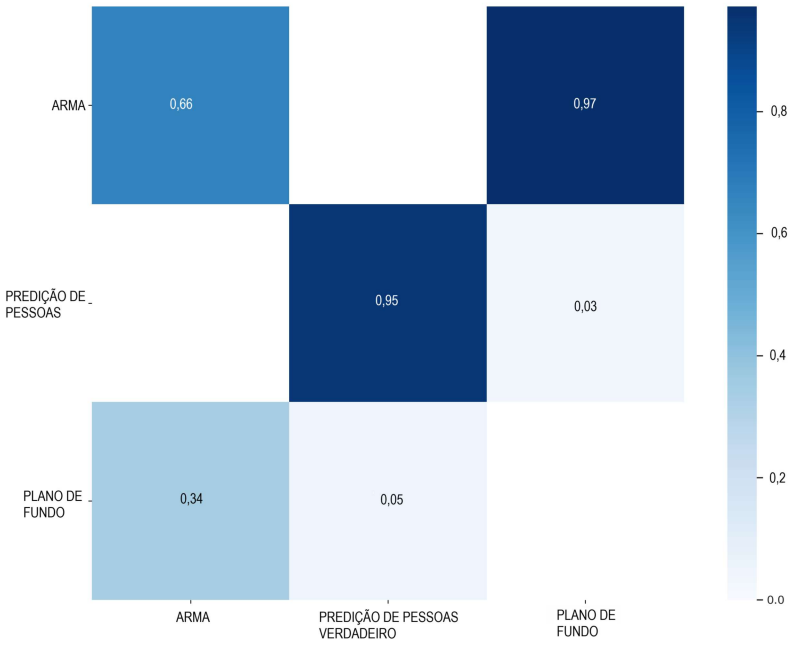
Primeiramente, é essencial comparar os resultados do treinamento do YOLOv8 para avaliar sua eficácia. Para essa comparação, foram utilizados dados da documentação oficial do YOLO fornecida pela *Ultralytics* disponível no *GitHub* [Jocher, Chaurasia e Qiu 2023].

Os parâmetros de perda, como 'box_loss' , 'val/box_loss', 'cls_loss', e 'val/cls_loss', apresentam valores baixos. Valores menores indicam que as caixas delimitadoras previstas estão bem alinhadas com as caixas verdadeiras, resultando em um desempenho melhor. Em contraste, os parâmetros de *recall* e precisão devem ser elevados, próximos de 1. Um bom *recall* indica que o modelo está capturando a maioria dos objetos nas imagens, enquanto uma alta precisão reflete a exatidão na identificação dos objetos, aumentando assim a confiabilidade do sistema.

Quanto aos valores de mAP, a documentação do YOLOv8 mostra que os valores durante o treinamento são um pouco superiores, mas próximos aos da rede padrão do YOLOv8, com um mAP de 44,9 na detecção usando o treinamento COCO. O valor indica que o modelo é eficaz em detectar e classificar objetos corretamente, o que mantém uma boa precisão em toda a gama de limiares de *recall*, sugerindo que o modelo é capaz de identificar e localizar objetos de forma eficiente.

Por fim, os parâmetros 'lr/pg' referem-se à melhor das *epochs* de treinamento da rede neural. De acordo com a documentação da *Ultralytics*, esses parâmetros são inicialmente configurados com valores baixos e ajustados ao longo das iterações do treinamento, pelo próprio algoritmo. Normalmente, o pg0 já é determinado pela própria rede e o pg1 está associado à taxa de aprendizado dos pesos das camadas YOLO. O pg2 representa a taxa de aprendizado para quaisquer parâmetros adicionais, como vieses. Começa-se com valores de 0,0001 para o pg1 e o pg2 e durante o processo de treino o algoritmo os ajusta até que se alcancem os melhores resultados de treinamento, momento em que esses valores se estabilizam.

Figura 27 – Matriz de confusão normalizada.



Fonte: Elaborado pelo autor.

A Figura 27 apresenta a matriz de confusão normalizada obtida durante a etapa de validação do modelo YOLOv8s. Cada linha da matriz representa a classe real dos objetos presentes nas imagens, enquanto cada coluna corresponde à classe predita pelo modelo. Os valores são proporções normalizadas por linha, ou seja, indicam a fração de objetos reais de uma determinada classe que foram atribuídos a cada classe predita. Os elementos na diagonal principal indicam os acertos do

modelo (classificações corretas), enquanto os valores fora da diagonal representam erros de classificação, podendo resultar, conforme o contexto, em falsos positivos ou falsos negativos. Ademais, observa-se que 97% das previsões classificadas como pertencentes à classe *gun* estavam, de fato, equivocadas, evidenciando uma confusão considerável do modelo, principalmente em relação à classe *person*.

Na classe *person*, o rendimento apresentou uma melhoria significativa. O modelo foi capaz de classificar com precisão 95% das amostras pertencentes a essa classe, evidenciando uma elevada taxa de acerto. Apenas 3% das amostras da classe *person* foram incorretamente categorizadas como *background*, e 5% das predições associadas à classe *person* foram errôneas, sendo classificadas como *gun*. Os resultados indicam que o modelo exibe elevada precisão para a classe *person*, a qual se revela a mais confiável dentre as três classes analisadas.

De maneira geral, o modelo demonstrou um desempenho satisfatório na identificação da classe *person*, entretanto, enfrenta consideráveis dificuldades ao classificar adequadamente amostras da classe *gun*, frequentemente confundindo-a com outras classes. Esse tipo de comportamento pode acarretar problemas em situações críticas nas quais a identificação correta de armas é fundamental. A análise indica que são imprescindíveis melhorias, principalmente com o intuito de aprimorar a precisão da classe *gun*. A precisão de cada classe foi calculada pelas fórmulas apresentadas abaixo:

- Classe *gun*:
  - TP = 66% (predições corretas de '*gun*').
  - P = contribuições erradas de '*gun*' vindas de '*background*' (97%).

$$\text{Precision\_}gun = \frac{0,66}{0,66 + 0,97} = 0,405 \tag{4.1}$$

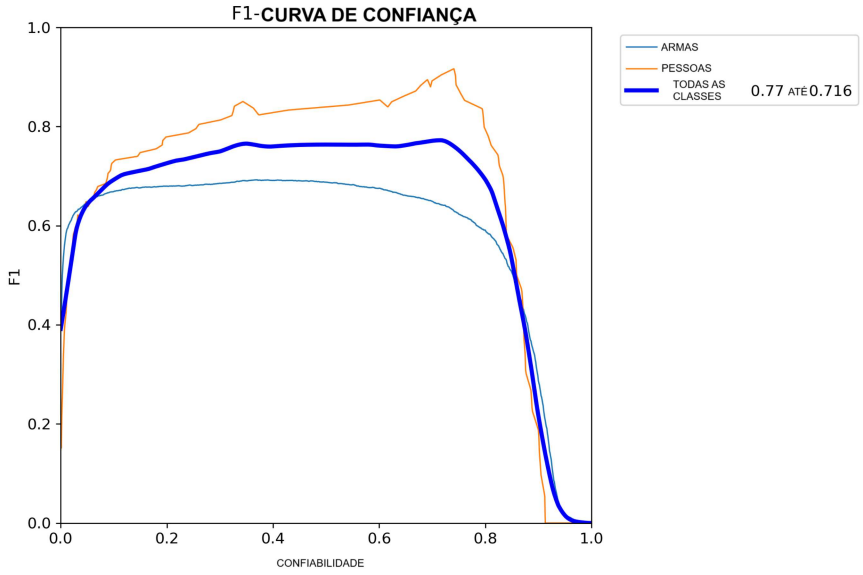
- Classe *person*:
  - TP = 95% (predições corretas de '*person*').
  - FP = contribuições erradas de '*person*' vindas de '*gun*' e '*background*' (baixas).

Precision_person é alta, estimada em mais de 0,9.

- Classe *background*:
  - TP = 3% (predições corretas de '*background*').
  - FP = contribuições erradas de '*background*' vindas de '*gun*' (34%).

$$\text{Precision\_}background = \frac{0,03}{0,03 + 0,34} \approx 0,081 \tag{4.2}$$

Figura 28 – Curva F1 - curva de confiança

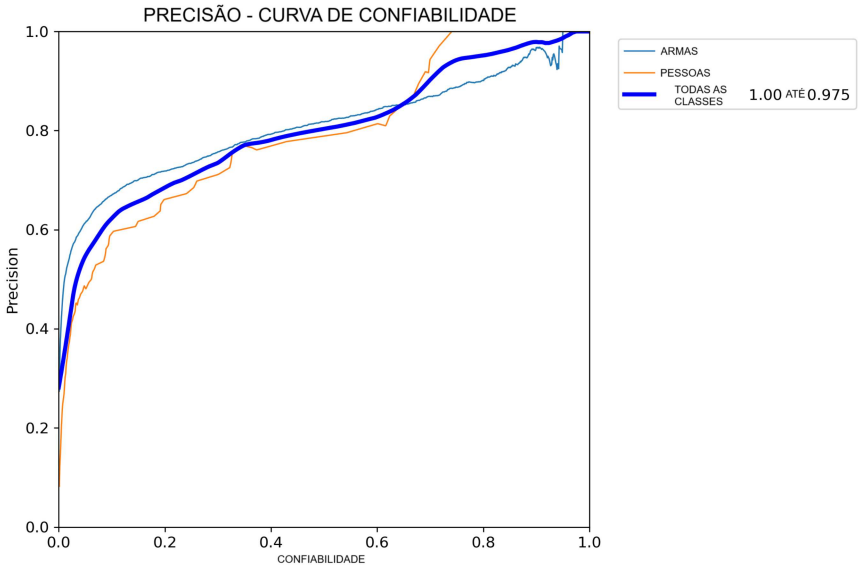


Fonte: Elaborado pelo autor.

O gráfico da Figura 28 apresenta a curva de confiança do F1-*score*, mostrando a relação entre a confiança nas previsões do modelo e seu desempenho. No eixo horizontal está o nível de confiança (0 a 1) e no vertical, o F1-*score* (0 a 1). Cada linha representa uma classe: azul clara para *gun*, laranja para *person* e azul espesso para o desempenho médio geral. O ponto destacado na curva azul indica F1-*score* máximo de 0,77 para confiança de aproximadamente 0,716.

A classe *person* apresenta desempenho superior à *gun*, mantendo F1-*score* mais alto em diversos níveis de confiança, o que evidencia maior facilidade do modelo em classificar pessoas corretamente. O F1-*score* atinge o máximo próximo de 0,7 de confiança, sugerindo que ajustar o limiar pode melhorar o desempenho global. Para níveis de confiança acima de 0,8, observa-se queda significativa no F1-*score*, indicando que o modelo se torna excessivamente rigoroso e exclui previsões potencialmente corretas.

Figura 29 – P curve - curva de confiabilidade.



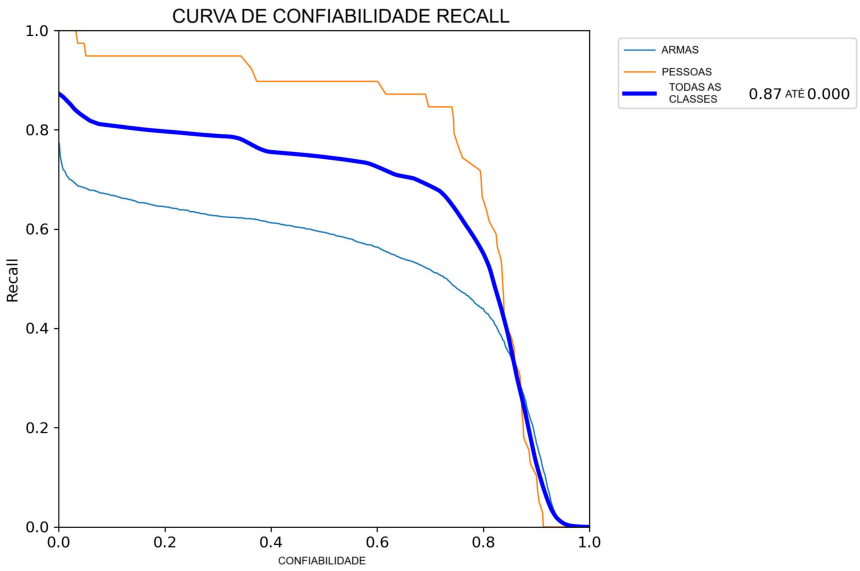
Fonte: Elaborado pelo autor.

O gráfico da Figura 29 apresenta a curva de confiança da precisão (*Precision-Confidence Curve*), mostrando como a precisão do modelo varia com o nível de confiança (0 a 1). Cada linha representa uma classe: azul clara para *gun*, laranja para *person* e azul espesso para a média global. O ponto destacado na curva azul indica precisão máxima de 1,00 para confiança de 0,975, evidenciando que previsões de alta confiança são extremamente precisas, embora em menor quantidade.

À medida que a confiança aumenta, a precisão também cresce, o que é esperado, mas isso pode reduzir o número de previsões, pois o modelo ignora exemplos abaixo do limiar. A curva da classe *person* apresenta pequenas oscilações,

mostrando variações na precisão ao longo do espectro de confiança. O ponto em que todas as classes atingem alta precisão indica um limiar eficaz para equilibrar precisão e cobertura, conforme a aplicação desejada.

Figura 30 – R curve - curva de confiabilidade de *recall*.

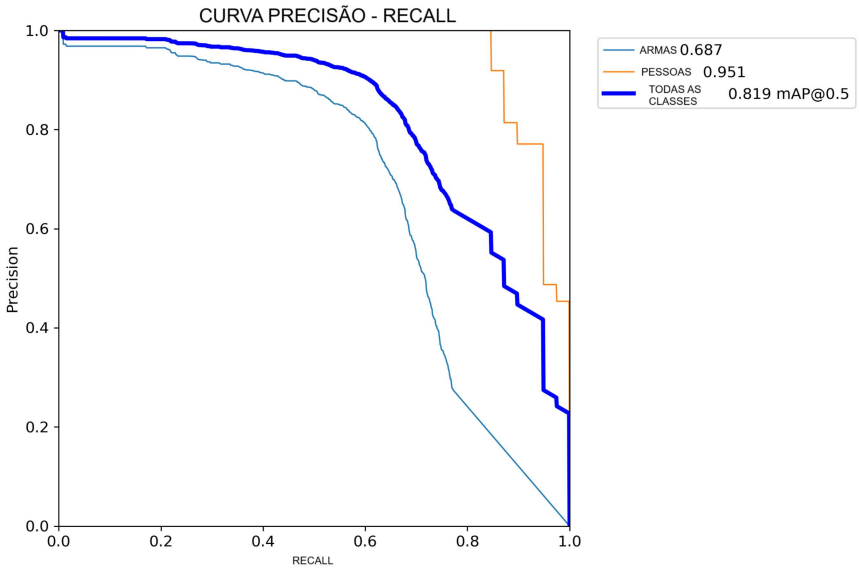


Fonte: Elaborado pelo autor.

O gráfico da Figura 30 apresenta a curva de confiança do *recall*, mostrando como o *recall* do modelo varia com o nível de confiança (0 a 1). Cada linha representa uma classe: azul clara para *gun*, laranja para *person* e azul espesso para a média geral. O ponto destacado na curva azul indica que, para confiança 0, o *recall* médio é 0,87, sugerindo que, ao aceitar todas as previsões sem considerar a confiança, o modelo recupera 87

À medida que a confiança aumenta, o *recall* diminui, pois previsões de alto grau de confiança podem desconsiderar instâncias corretas. A classe *person* mantém *recall* elevado na maior parte do intervalo, enquanto a classe *gun* sofre redução mais acentuada, indicando maior robustez na identificação de pessoas e maior sensibilidade da classe *gun* ao limiar de decisão. Para níveis de confiança próximos a 1, a queda acentuada do *recall* mostra que o modelo praticamente deixa de realizar detecções, o que pode não ser adequado dependendo da aplicação.

Figura 31 – PR curve - curva precisão e *recall*.

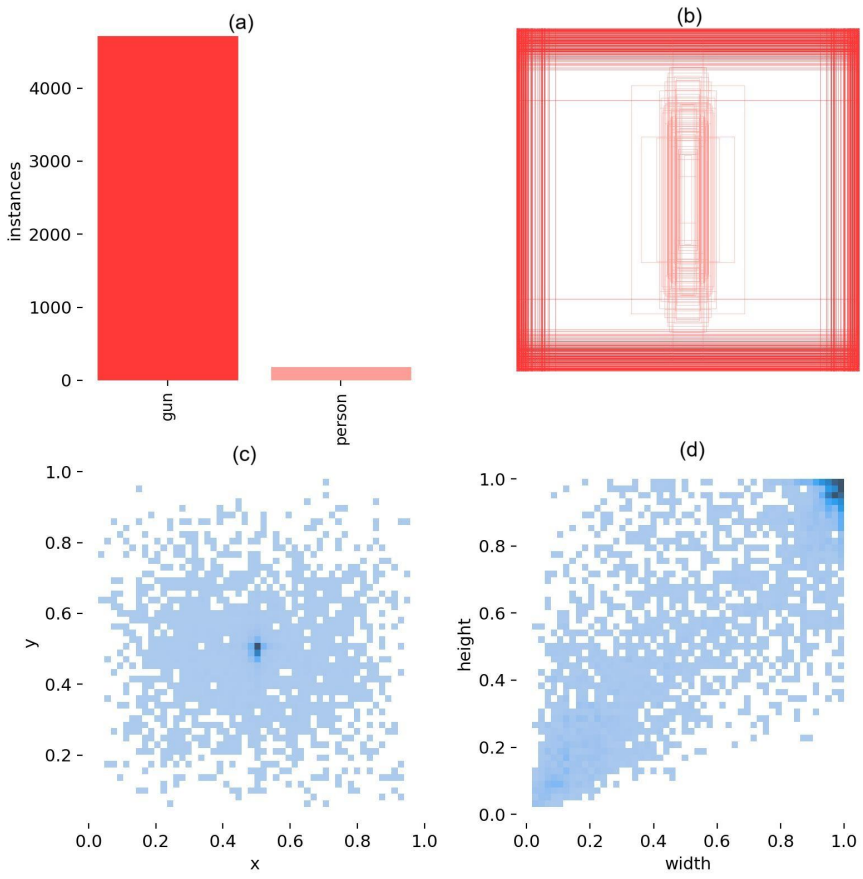


Fonte: Elaborado pelo autor.

A Figura 31 apresenta as curvas de Precisão-*Recall* das classes de interesse e um valor consolidado para todas as classes, com mAP de 0,819 para *IoU* 0,5. A curva ilustra a relação entre precisão, que mede a proporção de previsões corretas, e *recall*, que indica a proporção de amostras corretamente identificadas. A classe *person* apresenta desempenho superior, mantendo alta precisão e *recall* ao longo da faixa analisada, resultando em mAP de 0,951, sugerindo elevada eficiência devido a características visuais mais uniformes ou maior representatividade no conjunto de dados.

Já a classe *gun* mostra queda acentuada na precisão à medida que o *recall* aumenta, refletindo dificuldades do modelo em sustentar alta precisão em taxas elevadas de *recall*, com mAP de 0,687. A curva geral (linha azul espessa) resume o desempenho global, indicando resultados satisfatórios, mas apontando necessidade de melhorias, principalmente na classe *gun*. O gráfico evidencia a importância de equilibrar precisão e *recall* conforme o contexto da aplicação.

Figura 32 – Labels.



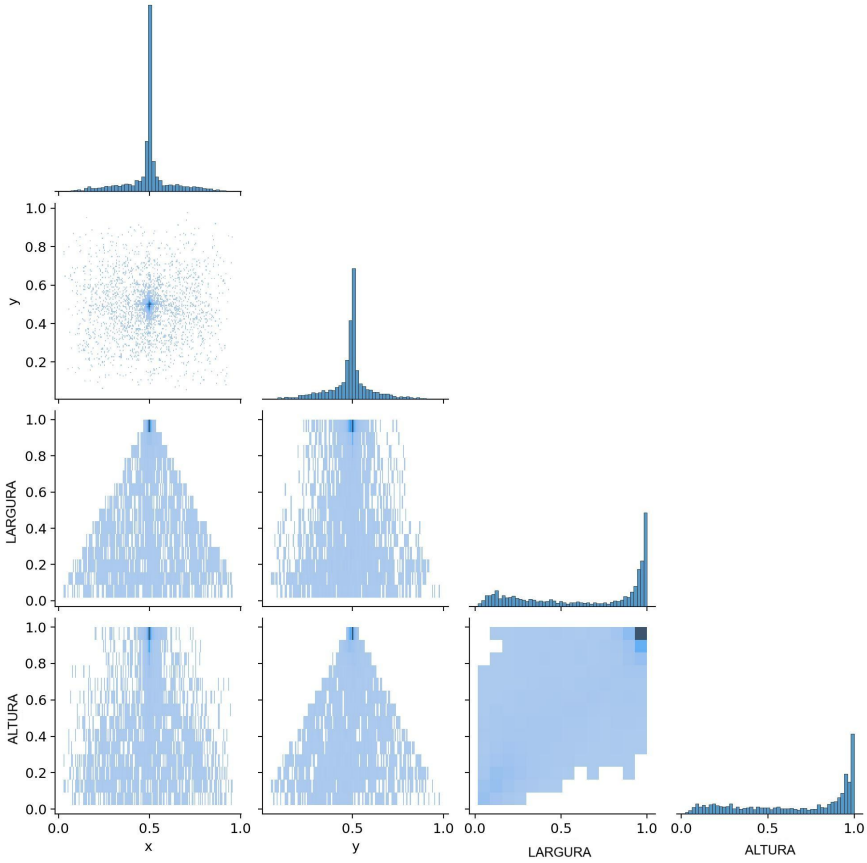
Fonte: Elaborado pelo autor.

Na Figura 32, observam-se quatro gráficos que demonstram informações estatísticas pertinentes aos dados de treinamento associados aos rótulos das classes *gun* e *person*. O gráfico (a) da Figura 32 ilustra a quantidade de instâncias por classe, evidenciando um notório desbalanceamento, uma vez que a classe *gun* apresenta um número de exemplos substancialmente superior ao da classe *person*. Já o gráfico (b) ilustra a distribuição das caixas delimitadoras no espaço de coordenadas. Observa-se que as caixas estão agrupadas na parte central, o que sugere que os itens apareceram de forma centralizada nas imagens.

Nos dois gráficos da parte inferior, são apresentadas as distribuições bidimensionais de características das caixas delimitadoras. No gráfico (c) da mesma

figura, encontra-se a representação da relação entre a posição x e y das caixas, evidenciando uma concentração no centro da figura. E o gráfico (d) apresenta a relação entre largura e altura das caixas, apresentando maior densidade em tamanhos reduzidos, o que pode sugerir que os objetos identificados frequentemente ocupam uma fração menor do quadro total.

Figura 33 – Correlação de legendas.



Fonte: Elaborado pelo autor.

Na Figura 33, é possível observar a correlação entre as principais características das caixas delimitadoras (x, y, largura e altura). Cada gráfico de dispersão une duas dimensões, possibilitando a identificação de correlações ou padrões entre elas. A concentração na diagonal das distribuições x e y representa o foco central dos objetos retratados nas imagens. A relação entre largura e altura também indica

que diversos objetos possuem dimensões proporcionalmente reduzidas. As distribuições marginais em cada um dos eixos, representadas por histogramas, corroboram essas análises, demonstrando que as caixas delimitadoras apresentam tamanhos diversos, porém com maior incidência em intervalos reduzidos. Esses critérios são fundamentais para compreender a maneira como os dados de entrada afetam o desempenho do modelo e contribuem para a otimização do treinamento, a fim de gerenciar de forma mais eficaz objetos menores ou posicionados centralmente.

Uma abordagem que pode apresentar resultados relevantes e aumentar a precisão deste trabalho é aplicar a metodologia de filtros de desfoque utilizada por Ashra *et al.*, (2022), que, ao implementar um filtro de desfoque Gaussiano conseguiu aumentar a precisão da rede, ao diminuir alguns dos índices, como contraste, nitidez e brilho. Em seu estudo foi utilizado o Yolov5, que apresentou relativa superioridade em relação às outras redes CNN disponíveis, mais precisamente sobre os valores de *recall* que foram apresentados. Ao aplicar o filtro de *blurr*, não houve expressiva alteração, mas no fim a precisão foi alterada como um todo, diminuindo os índices de falsos positivos e falsos negativos. Já no presente estudo, há uma aproximação no que diz respeito à precisão do modelo, sem a aplicação do filtro de *blurr* e, após a aplicação da lógica *fuzzy* os índices de alerta foram reduzidos de forma expressiva.

4.2 Resultado e discussão da análise dos vídeos

O modelo foi colocado em teste com a identificação dos vídeos de amostra (retirados do *YouTube*) em uma rede que não faz o envio de *frames* para o Telegram. Essa identificação teve como objetivo servir de método de estudo e embasar a afirmação de que em uma rede treinada, mas com baixo grau de confiabilidade, há alta incidência de identificações, sendo que muitas são falsos positivos, como apresentado na Tabela 7.

Tabela 7 – Desempenho do YOLOv8 treinado em diferentes vídeos sem ajustes.

Vídeo	***Frames***	**Incidência**
1	1520	454
2	1765	391
3	3229	50
4	5595	1442
5	9211	1838

Fonte: Elaborado pelo autor.

Foram testados diversos valores para o *treshold*. Inicialmente, utilizou-se 0,8, no entanto, observou-se que pouquíssimas detecções foram realizadas, o que, embora esperável, resultava em significativa perda de *frames* que continham objetos relevantes e que deixavam de ser contabilizados. Diante disso, o parâmetro foi ajustado até alcançar o valor de 'conf=0,47', que apresentou melhor desempenho na prática.

De maneira geral, aumentar o *threshold* torna o modelo mais criterioso, aceitando apenas predições com alto grau de certeza. Isso tende a aumentar a *precision*, pois reduz o número de falsos positivos (detecções incorretas). No entanto, pode resultar na perda de detecções verdadeiras, diminuindo o *recall*. Por outro lado, diminuir o *threshold* torna o modelo mais permissivo, o que aumenta o *recall*, pois mais objetos reais são identificados, mas também reduz a *precision*, devido à maior chance de ocorrerem detecções errôneas.

Tabela 8 – Desempenho do YOLOv8 treinado em diferentes vídeos com o *threshold* de 0,47.

Vídeo	***Frames***	**Incidência**	***Frames* salvos**	**Falsos positivos**
1	1520	112	23	5
2	1765	135	44	15
3	3229	22	12	3
4	5595	479	118	109
5	9211	492	135	110

Fonte: Elaborado pelo autor.

Com essa modificação, foi possível observar que a quantidade de incidência

para as classes de interesse sofreu uma grande redução, permitindo que os frames armazenados retratassem as situações em que as classes de interesse fossem identificadas.

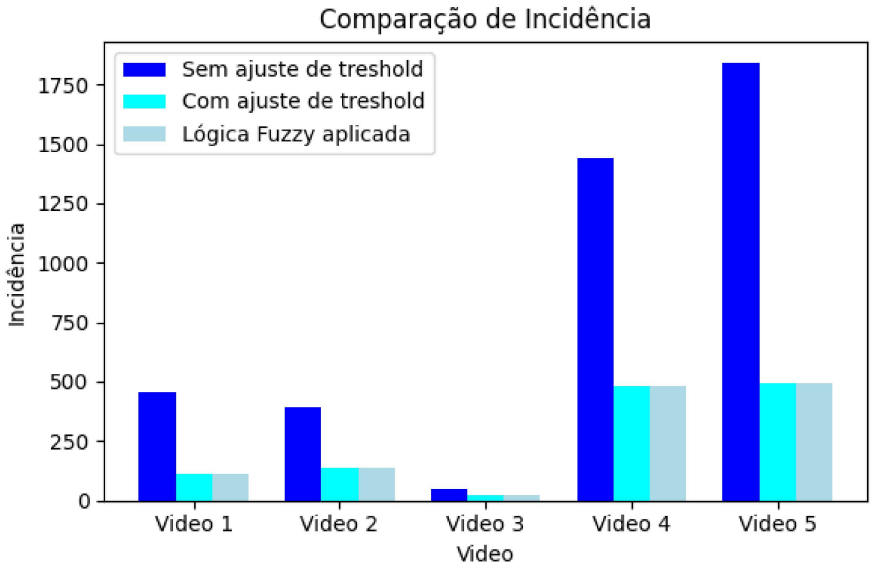
Tabela 9 – Desempenho do YOLOv8 treinado em diferentes vídeos junto da lógica fuzzy.

Vídeo	***Frames***	**Incidência**	***Frames* salvos**	**Falsos positivos**
1	1520	112	12	3
2	1765	135	22	7
3	3229	22	9	2
4	5595	479	47	6
5	9211	492	42	14

A Tabela 9 exibe os resultados alcançados após a implementação da lógica *fuzzy*. Em todos os vídeos analisados, observa-se uma diminuição na quantidade de *frames* capturados e na frequência de falsos positivos, em comparação ao modelo que funciona apenas com o ajuste de *threshold*. A redução observada é atribuída à aptidão do sistema fuzzy em avaliar as classes identificadas e as características da cena que está acontecendo, podendo classificar como uma situação de perigo de forma mais eficaz. Os resultados evidenciam que o sistema fuzzy opera de maneira eficaz na filtragem. Nos vídeos com maior duração, a disparidade é especialmente pronunciada: embora a soma total de incidências continue alta, a quantidade de falsos positivos diminuiu significativamente, o que sugere que o modelo foi mais preciso nas identificações.

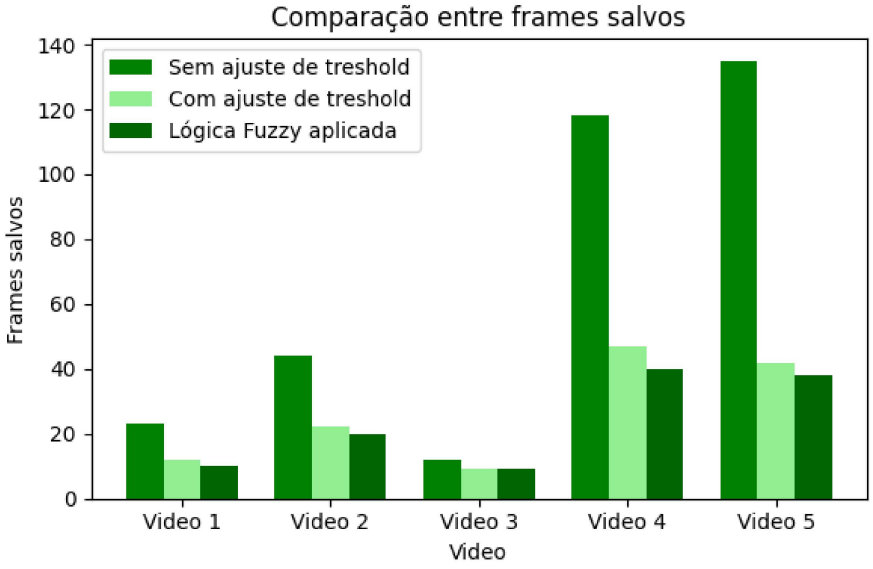
As Figuras 34, 35 e 36 apresentam, graficamente, a comparação dos dados das Tabelas 7, 8 e 9 frente ao *threshold* aplicado e as análises com as métricas *fuzzy*, o que reduz o número expressivo de incidências, de identificações das classes de interesse, o que permitiu maior confiabilidade ao sistema final.

Figura 34 – Comparativos de incidência.



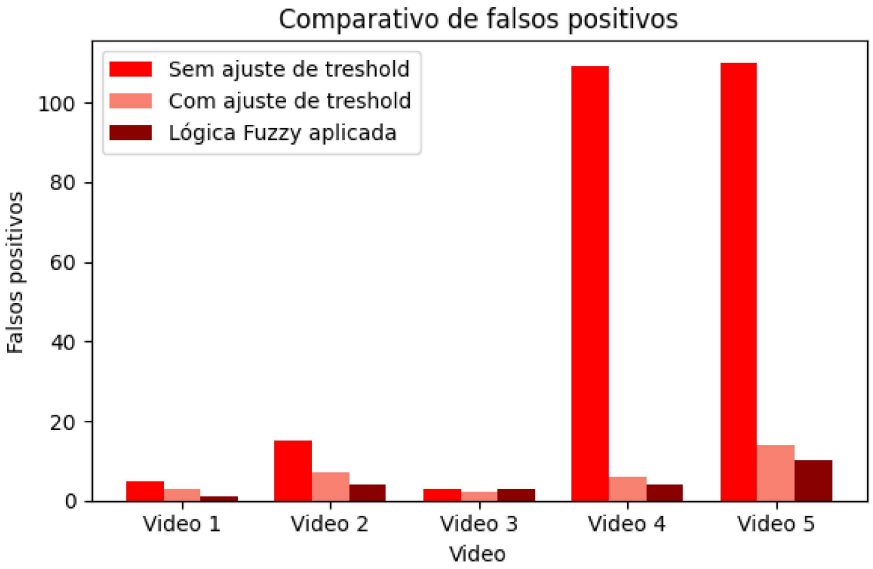
Fonte: Elaborado pelo autor.

Figura 35 – Comparativos de frames armazenados.



Fonte: Elaborado pelo autor.

Figura 36 – Comparativos de falsos positivos.



Fonte: Elaborado pelo autor.

Inicialmente, é importante destacar que todos os vídeos utilizados nos testes são oriundos de câmeras de segurança, com exceção do Vídeo 2, que foi capturado por uma câmera de ação acoplada ao capacete de um motociclista durante uma situação real de assalto em via pública. Esse vídeo foi intencionalmente selecionado por apresentar alta complexidade visual, com intensa movimentação, múltiplos elementos sobrepostos e grande variação de formas, o que coloca a identificação de objetos em uma situação de erros e propensão a apresentar muitos falsos positivos.

A duração dos vídeos impacta diretamente na quantidade de *frames* capturados para análise. Os vídeos 4 e 5, por serem significativamente mais longos, apresentaram naturalmente um maior número de quadros processados, resultando em maior incidência de detecções. Isso é refletido nos gráficos apresentados, especialmente no comparativo de incidência da Figura 34.

Na primeira etapa de filtragem, foi aplicado um ajuste no *treshold* baseado na pontuação de confiança das detecções. Observa-se uma redução significativa na quantidade de detecções, implicando diretamente em uma queda no número de *frames* salvos e, principalmente, na quantidade de falsos positivos gerados,

conforme ilustrado pela Figura 35. Essa etapa já representa uma melhoria relevante na precisão do sistema.

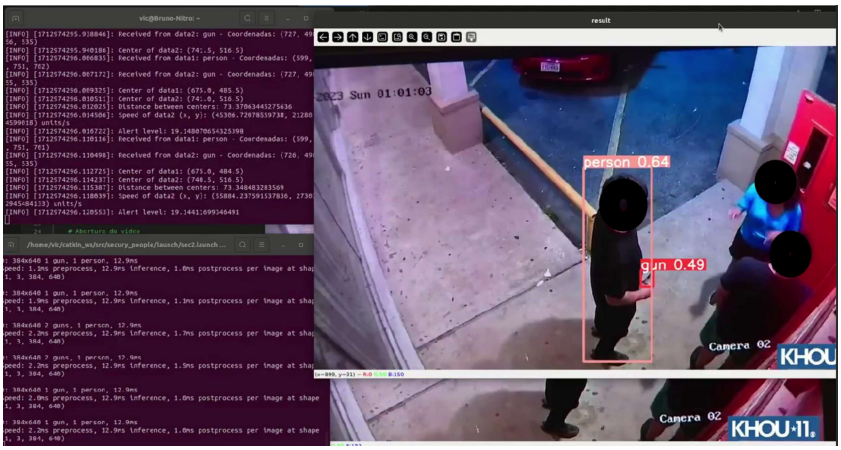
Na etapa seguinte, aplicou-se a lógica *fuzzy*, cujo objetivo não era apenas filtrar com base em uma taxa fixa de confiança, mas sim avaliar contextualmente a situação de risco da cena. Embora a redução na incidência geral não tenha sido tão expressiva em relação à etapa anterior, os gráficos revelam uma queda acentuada no número de *frames* salvos e de falsos positivos, indicando que a lógica fuzzy conseguiu refinar ainda mais os resultados, descartando detecções irrelevantes mesmo quando a pontuação de confiança era relativamente alta.

Essa redução nos *frames* salvos é particularmente relevante em sistemas de monitoramento, pois permite diminuir o uso de armazenamento e o custo computacional, mantendo a efetividade do sistema de segurança.

A Figura 36 apresenta o comparativo do número de falsos positivos identificados por vídeo nos três cenários avaliados. Nota-se que, na ausência de qualquer filtragem, os vídeos 4 e 5 produziram mais de 100 falsos positivos cada, o que representa um risco significativo para a operação confiável de sistemas de monitoramento. Com a introdução do limiar do *treshold*, observa-se uma queda acentuada desses valores, e, com a aplicação adicional da lógica *fuzzy*, os resultados são ainda mais refinados, especialmente em contextos visuais complexos, como no vídeo 2.

Mesmo após a aplicação da lógica fuzzy e o ajuste do threshold, ainda ocorreram falsos positivos na identificação. Isso pode ser observado na Figura 37, em que um objeto foi erroneamente identificado na mão de uma das pessoas na cena. Com essa identificação incorreta, os critérios para determinar uma situação de perigo — como baixa velocidade de movimentação da pessoa e do suposto objeto (arma), além da proximidade entre eles — foram atendidos, resultando em um falso positivo. No entanto, para que a lógica fuzzy efetivamente dispare um alerta, é necessário que a detecção ocorra em frames consecutivos, permitindo o cálculo da velocidade de movimento dos objetos. Como isso não aconteceu nesse caso, o alerta não foi gerado.

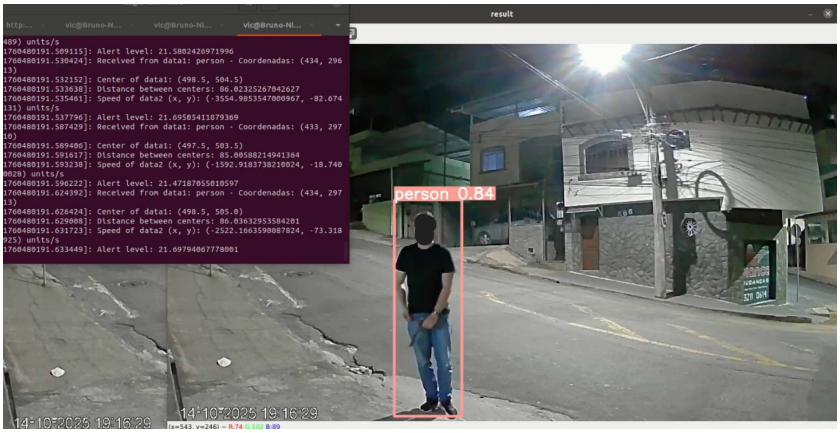
Figura 37 – Identificação errada que gerou um falso positivo.



Fonte: Elaborado pelo autor.

Outro ponto em que a detecção não foi tão eficiente, como demonstrado na Figura 38, em que houve uma sobreposição de imagens e a arma não foi identificada. Neste caso, é uma falha voltada ao treinamento do YOLO, que pode acabar gerando falsos negativos.

Figura 38 – Vídeo câmera erro sobreposição.

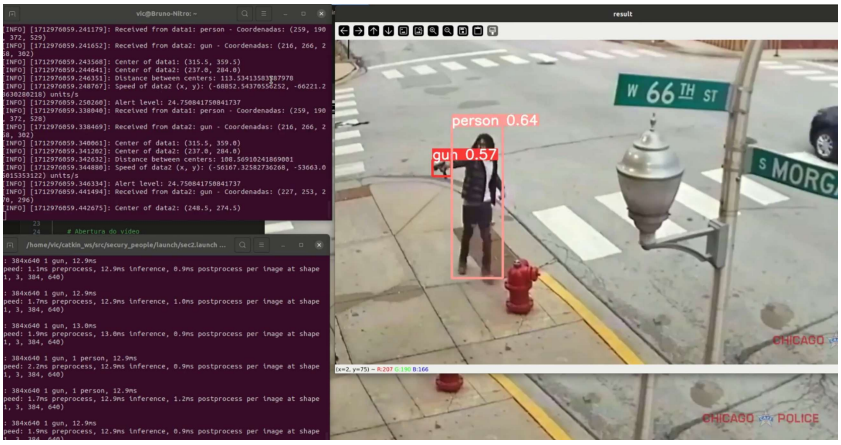


Fonte: Elaborado pelo autor.

Uma forma de diminuir esses alertas errados é fazer a contabilização de frames nos quais a classe foi identificada de forma sequencial: caso a classe não apareça em algum frame, esse alerta é removido e não é enviado; caso a classe

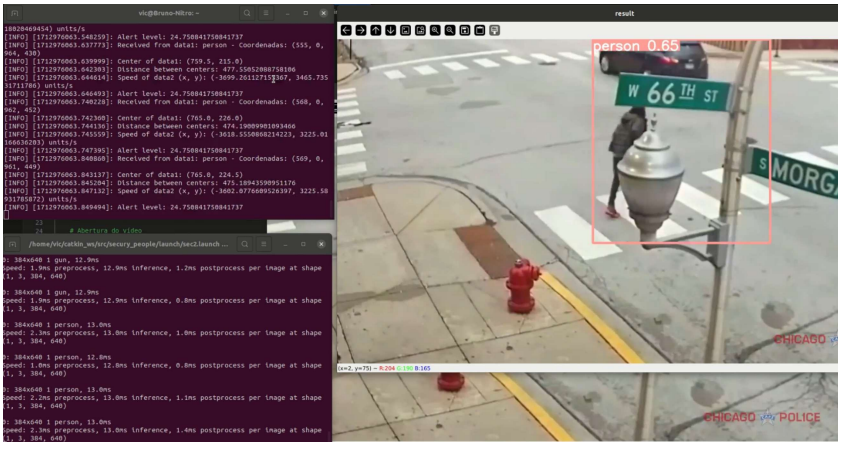
apareça em frames consecutivos, o alerta é gerado, o que foi demonstrado nas Figuras 39 e 40. Na Figura 39, a lógica *fuzzy* consegue determinar que é uma situação de perigo, exibindo o indicativo de alerta, pois as condições de baixa velocidade e distância entre objetos foram atingidas. Porém, poucos frames depois, na Figura40, houve a situação de fuga, mas, sem o indicativo de que havia uma arma no frame analisado, consequentemente, não foi gerado um novo alerta no terminal.

Figura 39 – Vídeo 5 situação de alerta



Fonte: Elaborado pelo autor.

Figura 40 – Vídeo 5 situação pós alerta



Fonte: Elaborado pelo autor.

Outro ponto a ser considerado é o envio de *frames* e mensagens no *bot* do Telegram. Para isso, foram propostos dois modelos: um síncrono e outro assíncrono. O modelo síncrono tende a apresentar travamentos, pois o fluxo de dados é limitado pela rede e pelo *bot*. Dessa forma, o programa tende a congelar por conta do envio em paralelo de várias imagens que estão sendo armazenadas em cache.

Já no modelo assíncrono, o envio é feito quando a rede o permite, sendo este um modelo que tende a apresentar maior estabilidade e melhores resultados durante a verificação de forma remota. Como cada mensagem enviada depende apenas da disponibilidade de memória, se o programa está ocupado, ele envia apenas que a classe de importância foi identificada, com informações de dia e hora; caso o programa tenha memória disponível, ele envia o *frame* com as informações de dia e hora da identificação.

Dessa forma, o modelo assíncrono foi o escolhido para a implementação do projeto aqui proposto, pois, além de permitir o envio sem o congelamento do sistema, permite que as identificações ocorram de forma mais fluida, sem deixar de enviar dados ao *bot* e ao algoritmo *fuzzy*. Essa implementação visa atender a uma demanda de menor especificidade, podendo ser implementada em sistemas de segurança residenciais. Dessa forma, os proprietários podem ter o controle das imagens, mesmo que de forma remota, sendo avisados caso ocorram situações potencialmente danosas às suas famílias ou aos bens das residências.

Em contextos comerciais, prediais ou empresariais é mais adequado implementar o alerta de identificação gerado pela lógica *fuzzy*, uma vez que, para um operador que trabalha fiscalizando diversas telas, esses alertas podem indicar momentos específicos de ameaça à segurança que podem passar despercebidos devido a lapsos atencionais.

5 Conclusões e Propostas Futuras

O presente trabalho teve como ponto de partida a constatação de que o crescimento acelerado dos centros urbanos, somado ao aumento da criminalidade e à popularização do uso de armas de fogo, impõe desafios significativos aos sistemas de segurança pública e privada. Diante desse contexto, torna-se essencial buscar soluções que aliem eficiência, acessibilidade e adaptabilidade, especialmente em relação ao monitoramento em tempo real por meio de câmeras. Dessa forma, a evolução dos sistemas de vigilância, potencializada por técnicas de visão computacional, leva a um caminho promissor para suprir as limitações dos métodos tradicionais.

Neste cenário, a integração entre o algoritmo de detecção de objetos YOLOv8 e a lógica *fuzzy* mostrou-se uma abordagem eficaz. A metodologia proposta foi combinar precisão técnica com aplicabilidade prática, utilizando recursos compatíveis com ambientes de baixo custo. Ao longo da pesquisa, buscou-se desenvolver uma arquitetura modular capaz de analisar vídeos reais, processar os dados em tempo real e emitir alertas automáticos com base em critérios de risco ajustáveis. O sistema foi validado com vídeos de câmeras de segurança e com uma câmera IP, refletindo situações próximas da realidade enfrentada em ambientes residenciais e comerciais.

Um dos pontos centrais da metodologia foi a definição de um limiar de confiança adequado para o processo de detecção. Inicialmente, valores mais altos, como 0,8, foram testados, mas resultaram em um número muito reduzido de detecções, uma consequência direta do rigor aplicado. Com isso, observou-se a perda de diversos *frames* potencialmente relevantes. Diante dessa limitação, optou-se por reduzir progressivamente esse valor até alcançar o *threshold* de 0,47. Esse ajuste permitiu manter um bom nível de precisão (89,85%), ao mesmo tempo em que aumentou significativamente a capacidade do modelo de reconhecer objetos relevantes em situações dinâmicas. Essa escolha estratégica refletiu uma preocupação em equilibrar *precision* e *recall*, priorizando a sensibilidade do sistema diante de possíveis ameaças.

Nesse ponto, é importante destacar que a escolha do *threshold* envolve uma decisão conceitual e prática: priorizar a precisão significa aceitar menos alarmes

falsos, mas também correr o risco de ignorar ameaças reais; já priorizar o *recall* significa detectar mais ocorrências, ainda que algumas sejam equivocadas. No caso de sistemas de segurança, como o proposto, é preferível errar por excesso de zelo do que por omissão, pois perder uma detecção crítica pode comprometer a segurança de pessoas e patrimônios. Assim, a decisão pelo valor de 0,47 foi embasada não apenas por critérios técnicos, mas também por um entendimento claro da aplicação prática e de suas implicações.

Outro diferencial relevante foi o uso da lógica *fuzzy*, que contribuiu de maneira significativa para o aprimoramento do sistema. Ao incorporar variáveis como distância e velocidade e interpretá-las com base em regras linguísticas adaptativas, o modelo passou a simular de forma mais próxima a lógica de tomada de decisão humana. Em contextos reais de vigilância, nos quais fatores como a movimentação e a proximidade de uma pessoa armada precisam ser avaliados de maneira flexível, essa abordagem mostrou-se extremamente útil. A lógica *fuzzy* permitiu a geração de alertas mais precisos e contextualizados, aumentando a confiabilidade e a robustez da solução proposta.

Com relação aos resultados obtidos, pode-se afirmar que os objetivos definidos no início do trabalho foram alcançados. O sistema foi capaz de identificar armas de fogo em diferentes condições e contextos, com desempenho satisfatório tanto em vídeos gravados quanto em transmissões pela câmera IP. A integração com o Telegram para o envio de alertas automáticos reforça a usabilidade do sistema, aproximando a tecnologia de seu propósito final: ser uma ferramenta de apoio à segurança em ambientes reais.

Embora os resultados sejam promissores, a pesquisa também abre caminho para diversas possibilidades de expansão. Uma delas é o aumento do escopo de detecção, incluindo armas brancas e outros objetos potencialmente perigosos. Além disso, a comparação do desempenho do modelo atual com versões mais recentes do YOLO, como o YOLOv11, pode fornecer novos *insights* e ganhos de desempenho. Outra frente interessante é o aprimoramento da interface com o usuário, oferecendo *dashboards* mais completos ou integração com sistemas maiores de segurança.

Por fim, destaca-se que a utilização de ferramentas como o YOLOv8 para identificação de objetos, aliada a técnicas como a lógica *fuzzy*, tem grande relevância

e potencial de impacto positivo para a sociedade. Sistemas como o proposto podem ser adaptados para diferentes contextos, contribuindo com a prevenção de incidentes, aumento da sensação de segurança e apoio a decisões estratégicas. Seja em ambientes residenciais, comerciais ou públicos, a tecnologia demonstrou que pode, de fato, atuar como aliada na construção de ambientes mais seguros e inteligentes.

REFERÊNCIAS

ALBEROLA, Á. M.; GALLEGO, G. M.; MAESTRE, U. G. *Artificial Vision and Language Processing for Robotics: Create end-to-end systems that can power robots with artificial vision and deep learning techniques*. [S.l.]: Packt Publishing Ltd, 2019.

ARAÚJO, A. M. et al. Detecção e destaque em vídeo de objetos utilizando yolo. Universidade Federal da Paraíba, 2022.

ARÇARI, A. Circuitos fechados de televisão: por uma estética do monitoramento e da vigilância. *Revista Farol*, v. 19, n. 29, p. 36–46, 2023.

ARNOLD, E. H. *Técnicas de aprendizado de máquina para detecção de pessoas em ambiente industrial*. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2016.

BACULI, A. L. et al. Uma nota sobre homicídios e a entrada de armas legais nas regiões brasileiras. *Revista Brasileira de Economia*, SciELO Brasil, v. 75, p. 1–14, 2021.

BALASUBRAMANIAN, V. et al. Detection and content retrieval of object in an image using yolo. *IOP Conference Series: Materials Science and Engineering*, v. 590, p. 012062, 10 2019.

BALDESSAR, G. et al. Detecção de armas de fogo em vídeo através de redes neurais. Florianópolis, SC., 2021.

BRAGE, L. B.; CAÑELLAS, A. J. C. Lógica difusa: una nueva epistemología para las ciencias de la educación. *Revista de educación*, v. 340, p. 995–1008, 2006.

BRAHMBHATT, S. *Practical OpenCV*. [S.l.]: Apress, 2013.

BRUNOWORKSPACE. Open Source Dataset, *Dataset-projeto Dataset*. Roboflow, 2025. <https://universe.roboflow.com/brunoworkspace-aauan/dataset-projeto>. Visited on 2025-06-28. Disponível em: <https://universe.roboflow.com/brunoworkspace-aauan/dataset-projeto>.

BUSTAMANTE, L. A.; GUTIÉRREZ, J. C. Real-time handgun detection using transformers on nvidia jetson agx xavier. In: IEEE. *2024 L Latin American Computer Conference (CLEI)*. [S.l.], 2024. p. 1–6.

COLODETTE, F. L. Sistema para o gerenciamento e monitoramento de tráfego de veículos. Cachoeiro de Itapemirim, 2023.

COSTA, T. K. O.; OLIVEIRA, M. V. L. d.; MARIANO, E. B. *Técnicas de segmentação de imagens através de redes neurais convolucionais*. Dissertação (B.S. thesis), 2023.

DESHPANDE, D. et al. Next-gen security: Yolov8 for real-time weapon detection. In: *2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. [S.l.: s.n.], 2023. p. 1055–1060.

EDUARDO, C.; VITO, E. L. D. et al. Introducción al razonamiento aproximado: lógica difusa. *Revista Americana de Medicina Respiratoria*, Asociación Argentina de Medicina Respiratoria, v. 6, n. 3, p. 126–136, 2006.

ELHARROUSS, O.; ALMAADEED, N.; AL-MAADEED, S. A review of video surveillance systems. *Journal of Visual Communication and Image Representation*, Elsevier, v. 77, p. 103116, 2021.

FAJNZYLBER, P.; JUNIOR, A. A. Violência e criminalidade. *Microeconomia e sociedade no Brasil*, Contra Capa Rio de Janeiro, p. 333–394, 2001.

GARCIA-GARCIA, B.; BOUWMANS, T.; SILVA, A. J. R. Background subtraction in real applications: Challenges, current models and future directions. *Computer Science Review*, Elsevier, v. 35, p. 100204, 2020.

GAT, A. et al. Animal detector system for forest monitoring using opencv and raspberry-pi. *IJRASET*, v. 10, p. 477–483, 2022.

GILLIS, A. Crime and state surveillance in nineteenth-century france. *American Journal of Sociology*, University of Chicago Press, v. 95, n. 2, p. 307–341, 1989.

GOMES, J. V. E. Detecção de objetos com a arquitetura yolo. 2022.

GOMES, R. M. V. O caso da vila dnocs, em sobradinho/df: a urbanização e o impacto na redução da criminalidade. UniCEUB, 2019.

GU, J. et al. Recent advances in convolutional neural networks. *Pattern Recognition*, Elsevier, v. 77, p. 354–377, 2018.

GULATI, S. et al. Paint/writing application through webcam using mediapipe and opencv. In: IEEE. *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*. [S.l.], 2022. v. 2, p. 287–291.

HARRISON, M. *Machine Learning–Guia de referência rápida: trabalhando com dados estruturados em Python*. [S.l.]: Novatec Editora, 2019.

HE, K. et al. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2961–2969.

IPEA. *MS/SVS/CGIAE - Sistema de Informações sobre Mortalidade*. Agressao com disparo arma fogo, Elaboração Diest/Ipea.

JAIN, A.; AISHWARYA; GARG, G. Gun detection with model and type recognition using haar cascade classifier. In: *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. [S.l.: s.n.], 2020. p. 419–423.

JOCHER, G.; CHAURASIA, A.; QIU, J. *Ultralytics yolov8. 2023*. 2023.

KANEHISA, R. F. de A. *Detecção de arma de fogo em imagens utilizando redes neurais convolucionais*. Dissertação (Monografia) — Universidade Federal do Maranhão, 2018.

KUNDETI, L. N. *Redefining Public Safety: A Comparative Analysis of RTDETR and YOLOv8–Unveiling The Future of Real-Time Handgun Detection*. Tese (Doutorado) — Dublin, National College of Ireland, 2025.

LI, Z.; HASEGAWA, A.; AZUMI, T. Autoware_perf: A tracing and performance analysis framework for ros 2 applications. *Journal of Systems Architecture*, Elsevier, v. 123, p. 102341, 2022.

LIMA, R. K. d.; MISSE, M.; MIRANDA, A. P. M. d. Violência, criminalidade, segurança pública e justiça criminal no brasil: uma bibliografia. Anpocs, 2000.

LIU, S. et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

LONG, D. et al. Ambient population and surveillance cameras: The guardianship role in street robbers' crime location choice. *Cities*, Elsevier, v. 115, p. 103223, 2021.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. *Estudos Avançados*, SciELO Brasil, v. 35, p. 85–94, 2021.

LYRA, S. et al. A deep learning-based camera approach for vital sign monitoring using thermography images for icu patients. *Sensors*, MDPI, v. 21, n. 4, p. 1495, 2021.

MACENSKI, S. et al. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, American Association for the Advancement of Science, v. 7, n. 66, p. eabm6074, 2022.

MAITY, M.; BANERJEE, S.; CHAUDHURI, S. S. Faster r-cnn and yolo based vehicle detection: A survey. In: IEEE. *2021 5th international conference on computing methodologies and communication (ICCMC)*. [S.l.], 2021. p. 1442–1447.

MARDEL, L. *Historia da arma de fogo portatil*. [S.l.]: Imprensa nacional, 1887.

MAYORAL-VILCHES, V.; CORRADI, G. Adaptive computing in robotics, leveraging ros 2 to enable software-defined hardware for fpgas. *arXiv preprint arXiv:2109.03276*, 2021.

MILANO, D. de; HONORATO, L. B. Visao computacional. *UNICAMP Universidade Estadual de Campinas FT Faculdade de Tecnologia*, 2014.

MORALES-LUNA, G. Introducción a la lógica difusa. *Centro de Investigación y Estudios Avanzados. México*, 2002.

MOREIRA, A. F. S.; SALES, A. C.; MOREIRA, C. S. Aspectos da privacidade na sociedade de vigilância: proteção de dados e sistemas de videomonitoramento. *Revista da EMERJ*, v. 26, p. 1–23, 2024.

MOSSELMAN, F.; WEENINK, D.; LINDEGAARD, M. R. Weapons, body postures, and the quest for dominance in robberies: A qualitative analysis of video footage. *Journal of Research in Crime and Delinquency*, Sage Publications Sage CA: Los Angeles, CA, v. 55, n. 1, p. 3–26, 2018.

MUHAMMAD, K. et al. Fuzzy logic in surveillance big video data analysis: comprehensive review, challenges, and research directions. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 54, n. 3, p. 1–33, 2021.

NGUYEN, D. et al. Transfer learning for abnormal object detection. *Journal of Technical Education Science*, v. 19, n. Special Issue 01, p. 25–32, 2024.

OLIVA, J. C. Aprendizado de máquina aplicado a um estudo de caso de classificação de qualidade de maçãs. Universidade Federal de Uberlândia, 2024.

OLIVEIRA, I. L. d. *Detecção de mamíferos em ambientes de mata utilizando YOLO*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2024.

PEGORARO, L. et al. Automated video monitoring of insect pollinators in the field. *Emerging topics in life sciences*, Portland Press Ltd., v. 4, n. 1, p. 87–97, 2020.

QI, D. et al. A dataset and system for real-time gun detection in surveillance video using deep learning. In: *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.: s.n.], 2021. p. 667–672.

RIBEIRO, H. M. C. et al. Aplicação do opencv utilizando técnicas de visão computacional e segmentação de imagens para reconhecimento de colônias bacterianas em análises microbiológicas de qualidade de água. *Caderno Pedagógico*, v. 21, n. 6, p. e4235–e4235, 2024.

ROBICHEAUX, P. et al. *RF-DETR: Um modelo SOTA de detecção de objetos em tempo real*. 2025. Blog Roboflow. Acesso em: 28 set. 2025. Disponível em: <https://blog.roboflow.com/rf-detr/>.

ROSA, A. C. F. d. et al. Uso de técnicas de aprendizado de máquina para classificação de fatores que influenciam a ocorrência de dermatites ocupacionais. *Revista Brasileira de Saúde Ocupacional*, SciELO Brasil, v. 48, p. e4, 2023.

SHEHZADI, T. et al. Object detection with transformers: A review. *arXiv preprint arXiv:2306.04670*, 2023.

SICSÚ, A. L. *Tecnicas de machine learning organização de Abraham Laredo Sicsu*. [S.l.]: Blucher, 2023.

SILVA, M. A. M.; CHAVES, A.; AQUINO, F. Projeto e desenvolvimento de uma ferramenta educativa para ensino de processamento de imagens baseado na biblioteca opencv. In: *XL Congresso Brasileiro de Educação em Engenharia, Belém*. [S.l.: s.n.], 2012.

SILVA, T. da. Visão computacional e racismo algorítmico: branquitude e opacidade no aprendizado de máquina. *Revista da Associação Brasileira de Pesquisadores/as Negros/as (ABPN)*, v. 12, n. 31, 2020.

SOARES, R. A. A. et al. Firearm detection using detr with multiple self-coordinated neural networks. *Neural Computing and Applications*, Springer, v. 36, n. 35, p. 22013–22022, 2024.

SON, J.; JUNG, H. Teacher–student model using grounding dino and you only look once for multi-sensor-based object detection. *Applied Sciences*, v. 14, n. 6, 2024. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/14/6/2232>.

SOO, S. Object detection using haar-cascade classifier. *Institute of Computer Science, University of Tartu*, v. 2, n. 3, p. 1–12, 2014.

TAN, M.; PANG, R.; LE, Q. V. Efficientdet: Scalable and efficient object detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. [S.l.: s.n.], 2020. p. 10781–10790.

VERMA, G. K.; DHILLON, A. A handheld gun detection using faster r-cnn deep learning. In: *Proceedings of the 7th international conference on computer and communication technology*. [S.l.: s.n.], 2017. p. 84–88.

WAGNER, M. et al. A learning robot for cognitive camera control in minimally invasive surgery. *Surgical Endoscopy*, Springer, v. 35, n. 9, p. 5365–5374, 2021.

YADAV, P.; GUPTA, N.; SHARMA, P. K. A comprehensive study towards high-level approaches for weapon detection using classical machine learning and deep learning methods. *Expert Systems with Applications*, Elsevier, v. 212, p. 118698, 2023.

ZAMBON, S. M. Aumento do efetivo policial militar x diminuição da criminalidade em guarapari, espírito santo1. *PRELEÇÃO*, 2012.

ZHAI, S. et al. Df-ssd: An improved ssd object detection algorithm based on densenet and feature fusion. *IEEE access*, IEEE, v. 8, p. 24344–24357, 2020.

ZHENG, J. et al. Extracting roadway background image: Mode-based approach. *Transportation research record*, SAGE Publications Sage CA: Los Angeles, CA, v. 1944, n. 1, p. 82–88, 2006.