UNIVERSIDADE FEDERAL DO PARANÁ

ADILSON CHRESTANI JUNIOR

ESTUDO DE CASO: AVALIAÇÃO DE MODELOS DE IDENTIFICAÇÃO E CONTAGEM DE PESSOAS POR FRAME EM AMOSTRAS DE VÍDEOS

CURITIBA, PR
2025

ADILSON CHRESTANI JUNIOR

ESTUDO DE CASO: AVALIAÇÃO DE MODELOS DE IDENTIFICAÇÃO E CONTAGEM DE PESSOAS POR FRAME EM AMOSTRAS DE VÍDEOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada no Programa de Pós-Graduação em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, da Universidade Federal do Paraná.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño.

CURITIBA, PR
2025

MINISTÉRIO DA EDUCAÇÃO
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PÓS-GRADUAÇÃO
CURSO DE PÓS-GRADUAÇÃO INTELIGÊNCIA ARTIFICIAL APLICADA - 40001016348E1

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INTELIGÊNCIA ARTIFICIAL APLICADA da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **ADILSON CHRESTANI JUNIOR**, intitulada: **ESTUDO DE CASO: AVALIAÇÃO DE MODELOS DE IDENTIFICAÇÃO E CONTAGEM DE PESSOAS POR FRAME EM AMOSTRAS DE VÍDEOS**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 18 de Junho de 2025.

RAZER ANTHOM NIZER ROJAS MONTANO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Estudo de caso: Avaliação de modelos de identificação e contagem de pessoas por frame em amostras de vídeos

Adilson Chrestani Junior
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná
Curitiba, Brasil
adilson.chrest@gmail.com

Prof. Dr. Razer Anthom Nizer Rojas Montaño
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná
Curitiba, Brasil
razer@ufpr.br

***Resumo*—Este relatório apresenta uma possível solução para a identificação de colaboradores (humanos) em uma linha de produção de uma fábrica do segmento de Cuidado e Beleza. Através da correta identificação dos funcionários e suas respectivas posições geográficas em uma indústria, é possível trabalhar tanto em projetos de otimização de mão de obra quanto de segurança. O relatório analisou 4 modelos de detecção de pessoas - Haar Cascade, HOG, YOLO v8 e CSRnet - aplicados em uma amostra de 5 vídeos com o intuito de contar a quantidade de pessoas por frame. Com o objetivo de analisar o caráter generalista dos modelos, foram consideradas as mesmas operações de pré-processamento para as famílias dos modelos, mudando apenas o valor dos parâmetros utilizados em cada cenário. Considerando os 60 cenários de resultados, o algoritmo com melhor *trade-off* entre assertividade na contagem de pessoas e performance de execução é o *YOLO modelo_tuned*, sendo capaz de obter a melhor qualidade na contagem de pessoas por frame de vídeo de todos os modelos - através das métricas de MAE, MSE e RMSE - ao passo que manteve um tempo de processamento médio razoável.**

***Palavras-chave*—visão computacional, hog, haar cascade, yolo, csrnet, processamento de imagem**

***Resumo*—This report presents a feasible solution for identifying workers (humans) on a production line in a factory within the Care and Beauty segment. Through the correct identification of employees and their respective geographic positions within an industry, it is possible to work on both labor optimization and safety projects. The report analyzed four people detection models — Haar Cascade, HOG, YOLO v8, and CSRnet — applied to a sample of five videos with the purpose of counting the number of people per frame. To guarantee the generalist nature of the models, the same preprocessing operations were applied across the model families, changing only the parameter values used in each scenario. Considering the 60 result scenarios, the algorithm with the best trade-off between accuracy in people counting and execution performance is the YOLO tuned model, which achieved the highest quality in counting people per video frame among all models — based on the MAE, MSE, and RMSE metrics — while maintaining a reasonable average processing time.**

***Index Terms*—computer vision, hog, haar cascade, yolo, csrnet, image processing**

I. DESENVOLVIMENTO

Na década de 80 foi definido por Ballard e Brown [1] que a Visão Computacional é a ciência que possibilita a criação de tecnologias capazes de dar autonomia às máquinas para a extração e compreensão do meio através da imagem. Recentemente, o tópico entrou em alta por conta do aumento exponencial da disponibilidade de imagens para bases de treinamento, causado pela popularização dos celulares e redes sociais, uma melhora da capacidade de processamento dos computadores e um maior foco acadêmico em soluções para aumentar a acuracidade dos modelos e seu tempo de execução [2].

Dentre as inúmeras tarefas nas quais é possível replicar a visão computacional, pode-se destacar a detecção de pedestres, placas e condução dos carros na direção autônoma [3], o auxílio no diagnóstico de doenças através de exames de tomografias, ressonâncias e ultrassons na área médica [4] e no ambiente fabril a identificação de potenciais riscos de acidentes ou riscos ergonômicos, na visão do colaborador, e nas estações de verificação que aferem posições de montagem e qualidade das peças produzidas [5].

De acordo com Gonzalez e Richard embora não haja uma separação exata entre o que faz parte da área de processamento de imagem e de visão computacional, é bem vindo enxergar em uma linha contínua que os processos se dividem em três tipos: níveis baixo, médio e alto. A etapa de nível baixo concentra operações primitivas de pré-processamento - realce do contraste, remoção de ruídos e aguçamento da qualidade da imagem. O nível médio abrange as operações de segmentação de *features*, que consiste em obter os atributos morfológicos extraídos delas. O último nível envolve dar sentido a um conjunto de objetos, ou *features*, reconhecidos na etapa anterior, realizando a função cognitiva geralmente associadas à visão [6].

Apesar de uma tendência de crescimento geral da Inteligência Artificial na indústria, o segmento de visão computacional tem apresentado um destaque acima da média nos últimos anos. Embora tais sistemas de visão já sejam parte da arquitetura de muitas máquinas em uma linha de produção, são poucos os materiais que se aprofundam na tecnologia e hardwares de visão computacional em termos de indústria [4] - embora não seja explicitamente citado no texto, possivelmente muita desta falta de material é influenciada por acordos de *non-*

disclosure agreement (N.D.A) que visam proteger segredos industriais responsáveis pelos bons resultados das tecnologias implementadas. Voltado para o ambiente fabril, Grazielle *et. al.* [7] propôs a implementação de um sistema de Visão Computacional em um manipulador robótico industrial através de um *raspberry* com o objetivo de identificar a posição das peças.

Este trabalho conduziu um estudo de caso de soluções atuais de algoritmos de detecção de seres humanos com o objetivo futuro de implementação de um sistema de contagem de mão de obra em uma linha de produção de uma fábrica de cosméticos. Para fins de privacidade de dados, porém, toda a análise deste relatório foi conduzida a partir de vídeos de domínio público. Seguindo a estrutura proposta por Gonzalez e Woods [6], os algoritmos foram divididos em duas partes: pré-processamento das imagens em *streaming* e contagem dos humanos presentes nelas através de quatro métodos: *Haar Cascade*, *HOG*, *Yolo V8* e *CSRNet*.

A. Descrição dos dados

Os vídeos utilizados neste estudo de caso foram obtidos a partir do bancos de dados online [8]. Todos os vídeos deste relatório estão disponibilizados em domínio público.

Por se tratar de um estudo exploratório acerca do potencial de identificação e contagem de humanos por *frame* em vídeos de multidão, foram utilizadas 5 gravações extraídas em contextos e resoluções distintas, com o intuito de avaliar o caráter generalista de cada um dos quatro modelos selecionados.

Figura 1. Frame retirado do primeiro vídeo (VD1) do estudo
Fonte: O autor (2025)

Para cada vídeo, foram geradas máscaras que definem para o algoritmo zonas de interesse onde a identificação e contagem dos seres humanos devem ocorrer. Na figura 6 tem-se um exemplo de recorte gerado pela máscara do vídeo VD1. A aplicação de máscaras ajuda a reduzir significativamente o tempo de inferência pois o modelo foca apenas nas regiões relevantes da imagem [9].

B. Métodos

Esta seção apresenta os métodos utilizados para detecção e contagem de humanos nos cinco respectivos objetos de estudo.

Figura 2. Frame retirado do segundo vídeo (VD2) do estudo
Fonte: O autor (2025)

Figura 3. Frame retirado do terceiro vídeo (VD3) do estudo
Fonte: O autor (2025)

Antes da implementação do algoritmo de identificação de humanos, é necessário trabalhar no pré-processamento das imagens de forma a otimizar os resultados finais e tempo de treino do modelo. É possível reduzir em até 4 vezes o tempo de treinamento através de métodos tradicionais de alteração morfológica da imagem como Normalização e Equalização por Histograma [10].

Para cada modelo de Inteligência Artificial (IA), foi apresentado um detalhamento do seu racional, o método de pré-processamento de *frames* utilizado e sua implementação em *Python*.

Figura 4. Frame retirado do quarto vídeo (VD4) do estudo
Fonte: O autor (2025)

Figura 5. Frame retirado do quinto vídeo (VD5) do estudo
Fonte: O autor (2025)

Figura 6. Exemplo de máscara utilizada no (VD1) na hora de avaliar um zona específica para a contagem de pessoas
Fonte: O autor (2025)

1) Método Haar Cascade: O algoritmo de Viola-Jones [11], representa uma das primeiras abordagens viáveis para detecção rápida e em tempo real de objetos em imagens, especialmente rostos humanos. Essa técnica introduziu três contribuições principais: *Haar-like features*, Imagens integrais e *Adaboost*.

Haar-like features são características extraídas a partir de filtros *Haar-like* que funcionam como kernels convolucionais para detecção de bordas em imagens. A diferença de kernels tradicionais é que as características *Haar-like* se assemelham à traços humanos, como rostos e bocas - tendo, inclusive, a habilidade de mudar a sua escala para detectar faces maiores ou menores das imagens que compuseram o classificador [12]. Estas saídas são computadas através de imagens integrais, método criado por [11] que tem como objetivo otimizar o trabalho de identificação das características *Haar* na imagem ou *frame* de vídeo.

Ao considerar-se uma janela de detecção de 24x24 *pixels*, obtem-se um conjunto de mais de 160 mil vetores *Haar-like* [11]. A técnica de *Adaboost* tem como objetivo agrupar vetores de características de forma tanto a criar classificadores fortes que combinem diferentes *Haar-features* quanto otimizar o tempo de processamento ao descartar classificadores fracos. Historicamente, o *Adaboost* era muito utilizado em algoritmos de árvores de regressão [12], todavia, com as contribuições de Viola [11] passou a ser uma ferramenta importante na classificação em imagens. Uma vez com as melhores características definidas pelo *Adaboost*, o processo de classificação acontece em uma cascata de validações na imagem a procura destas características fortes, dando nome ao método *Haar Cascade*.

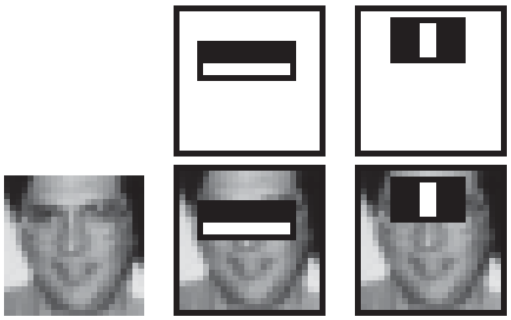

Figura 7. Exemplos de filtros *Haar-like* implementadas no modelo *Haar Cascade*
Fonte: Paul Viola (2001)

Como etapa de pré-processamento para a implementação do *Haar Cascade*, foram utilizadas as técnicas de escala cinza e filtro de Medianas para a remoção de ruídos [13]. Além dos tratamentos das imagens dos *frames*, também implementou-se uma máscara que delimita uma região de interesse no *frame* para focar a contagem de pessoas do modelo.

A implementação do *Haar Cascade* foi dada através da biblioteca OpenCV. O termo *Haar Cascade Classifier* refere-se a uma implementação prática do algoritmo Viola-Jones, baseada em *Haar-features* treinado previamente para detectar classes específicas como rostos, corpos ou olhos [11]. Pelo OpenCV, tem-se um modelo em xml das *Haar features* mais fortes, o que encurta o tempo de implementação da solução ao não precisar treinar estes classificadores para obter seus pesos em cima de uma base de dados. A ordem de tarefas do modelo *Haar Cascade* empregado neste relatório é encontrada na sequência:

1) Inicializa o modelo já treinado do *Haar Cascade*;
2) Inicializa vídeo a ser analisado;
3) Aplica o pré-processamento da imagem do *frame*;
4) Em cada *frame* é aplicada a função *detectmultiScale*: A imagem é dividida em zonas, ou janelas, de tamanho delimitado pelo parâmetro *minsize*. Em cada janela, são analisados os diferentes estágios da cascata *feature Haar* para avaliar se há características visuais semelhantes às de uma pessoa. Se esta janela passar por todos os classificadores do modelo treinado, temos uma detecção positiva. Após estas detecções, é realizado uma sobreposição das janelas pelo parâmetro *minneighbors* que valida a partir de quantas janelas agrupadas a detecção é válida. Por fim, todas as detecções positivas são agrupadas para o resultado de pessoas por *frame* de vídeo.

A fácil implementação do algoritmo *Haars Cascade* atrelado aos bons resultados encontrados na bibliografia [12] o tornam um excelente solução para aplicação de identificação de pessoas em frames de vídeo. O ponto negativo, porém, está

na limitação nos ajustes de parâmetros - O *Haars Cascade* nativo do OpenCV possui poucas possibilidades de alteração de parâmetros para uma melhor identificação do modelo.

2) Método HOG: O método de Histograma de Gradientes Orientados (HOG) consiste em um modelo composto por duas etapas: A *feature extraction* através do Histograma de Gradientes Orientados para extração dos histogramas representativos do respectivo objeto e a utilização destes vetores de *features* em um classificador por Máquina de Vetores de Suporte (SVM) [14].

Para a geração destes gradientes orientados, a imagem é dividida em pequenas células de *pixels* onde são calculados os valores de magnitude e direção do gradiente para àquele espaço. Na sequência tem-se a normalização, ou agrupamento de vizinhos, que tem como objetivo suavizar variações bruscas de iluminação e contraste. Por fim, é calculado um Histograma considerando a magnitude das células da imagem no eixo x e a direção do vetor no eixo y [15]. Na figura 8 pode-se observar um mapa visual dos vetores orientados que refletem as bordas principais do objeto.

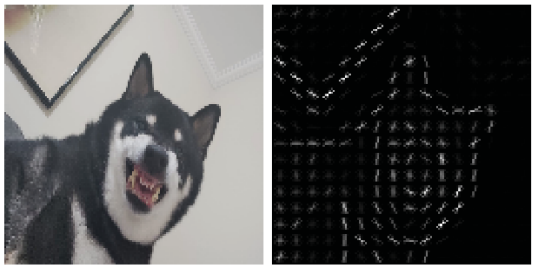

Figura 8. Representação de uma foto utilizando os vetores de gradiente orientado
Fonte: O autor (2025)

Um SVM é um modelo de Aprendizado de Máquina capaz de realizar tanto classificações lineares quanto não lineares [16]. No caso do modelo *HOG*, é alimentado ao SVM os respectivos histogramas de características das imagens com uma *label*. O próprio método de aprendizado de máquina é capaz de delimitar as margens que separam cada classe, criando assim um classificador eficiente de presença ou não do objeto de estudo [14].

Como etapa de pré-processamento para a implementação do *HOG*, foi-se aplicada a técnica de escala cinza, uma vez que é um requisito para o gradientes de intensidade [15], uma Equalização de Histograma Adaptativa por Contraste (CLAHE) para melhora do contraste e uma máscara que delimita uma região de interesse no *frame*. A ordem de tarefas do modelo *HOG* empregado neste relatório pode ser encontrada na sequência:

1) Inicializa o modelo já treinado de SVM utilizando como *input* o *HOG*;
2) Inicializa vídeo a ser analisado;
3) Aplica o pré-processamento da imagem do frame;
4) Em cada frame aplica-se uma a função *detectmultiScale*: A imagem é dividida em janelas de tamanho delimitado pelo parâmetro *winstride*. Em cada janela, é calculado o *HOG* e alimentado ao SVM - caso o valor esteja dentro do estipulado na pontuação de confiança do modelo de aprendizado de máquina, marca-se uma detecção na janela em específico. Além da janela, a avaliação da detecção é realizada em diferentes escalas de imagem e pode ser configurada pelo parâmetro *scale*. No final, as detecções positivas para humanos são somadas, gerando o valor de pessoas por *frame* de vídeo.

Assim como o algoritmo *Haar Cascade*, o ponto negativo da implementação no *OpenCV* do método *HOG* está na limitação nos ajustes de parâmetros, uma vez que a biblioteca possui possibilidades limitadas, principalmente no modelo SVM pré-treinado empregado.

3) Método YOLO: O modelo YOLO *(You Only Look Once)*, introduzido por Redmon *et al.* [17], representa uma abordagem inovadora para a detecção de objetos. Esta técnica se destaca por sua capacidade de realizar detecção em tempo real ao tratar o problema como uma única tarefa de regressão, prevendo diretamente as caixas delimitadoras e as probabilidades de classe de cada objeto presente na cena. Diferente dos métodos tradicionais abordados neste relatório como HOG ou *Haar Cascade*, que baseiam sua detecção em extração manual de características locais e múltiplas etapas de classificação e filtragem, o YOLO trata tudo isso de forma unificada.

Uma das características fundamentais do YOLO é a sua estrutura inspirada em redes de classificação, como o GoogLeNet, adaptada para a tarefa de detecção. O modelo utiliza camadas convolucionais para extrair características hierárquicas e generalizáveis das imagens, sendo capaz de processar *frames* completos de vídeo sem a necessidade de algoritmos adicionais de busca de regiões de interesse [18]. Com o mapa de características extraído, o YOLO divide a imagem de entrada em uma grade e confronta cada segmento com o mapa de características, prevendo para cada uma das células a criação de um vetor contendo informações posição e tamanho da *bounding box*, score de confiança e probabilidades de classes. [19].

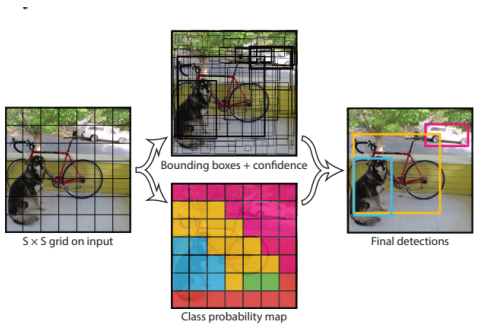

Figura 9. Representação de uma foto utilizando os vetores de gradiente orientado
Fonte: Lingala [17] (2016)

São detectadas diferentes *bounding boxes* para o mesmo objeto, conforme Figura 9. A forma encontrada por [17] para definir o contorno com maior probabilidade de englobar o

objeto foi através do algoritmo *Non-Maximum Suppression* (NMS): Para cada objeto, o NMS seleciona a caixa com maior score de confiança, calculando o *Intersection over Union* (IoU) e descartando as com previsões de baixa assertividade, garantindo o modelo uma única caixa por objeto.

Como etapa de pré-processamento para a implementação do YOLO v8, foi-se aplicada a técnica de escala cinza e uma Equalização CLAHE para aumento no detalhamento dos *frames* dos vídeos mais afastados, conforme realizado por Chen *et al.* [20].

A ordem de tarefas do modelo YOLO v8 empregada neste relatório pode ser encontrada na sequência:

1) Inicializa o modelo de detecção *yolov8m* já treinado;
2) Inicializa o *tracker* com os seguintes parâmetros: *max_age* que define o número de *frames* que um objeto pode sumir antes de ser descartado, *min_hits* que estabelece algumas detecções mínimas antes de validar o objeto e *iou_threshold* que associa detecções à rastreamentos;
3) Inicializa vídeo a ser analisado;
4) Aplica o pré-processamento da imagem do *frame*;
5) Para cada *frame*: É aplicada a função *model* para executar a detecção YOLO na região de interesse. A função possui três parâmetros muito similares aos do *tracker*: *stream* para processos iterativos, *conf* que define a confiança mínima para considerar uma detecção e *iou* para supressão de não-máximos, garantindo uma única caixa por objeto detectado;
6) Para cada detecção: Extrai *bounding box*, confiança e classe, filtrando no caso apenas o índice 0 referente à detecção de humanos, passa as detecções para o *tracker* e conta quantas pessoas foram detectadas e rastreadas no frame

Outro aspecto relevante do YOLO é a sua eficiência computacional e adaptabilidade a diferentes contextos. Embora Wang [19] tenha observado que o modelo tende a apresentar dificuldades na detecção de pequenos objetos, sua estrutura compacta e o uso de uma única passagem pela rede tornam o YOLO ideal para detecção em tempo real.

4) Método CSRnet: O CSRNet, proposto por Yuhong [21] é uma rede neural convolucional (CNN) desenvolvida especificamente para a contagem de multidões e geração de mapas de densidade de alta qualidade em frames altamente congestionados. Conforme descrito no artigo, o CSRNet é composto por duas partes principais: um *front-end* baseado na VGG-16 - com as camadas totalmente conectadas removidas - para a extração de características bidimensionais e um *back-end* que utiliza camadas convolucionais dilatadas. Segundo o autor, estas camadas permitem a expansão do campo receptivo sem o uso de operações de *pooling*, o que preserva a resolução espacial da imagem e melhora a representação da densidade do público.

A abordagem de Yuhong [21] resolve algumas deficiências dos métodos tradicionais de contagem de multidões, como o excesso de ramificações em MCNN - que introduzem redundância e sobrecarga computacional. Ao evitar o uso de classificadores de nível de densidade (presentes, por exemplo, no Switching-CNN e no CP-CNN), a CSRNet elimina a dependência de pré-classificações, reduzindo o número de parâmetros e melhorando a generalização. O segundo diferencial técnico é a adoção de convoluções dilatadas para substituir *pooling*, permitindo ao modelo manter resolução espacial enquanto capta informações contextuais de múltiplas escalas.

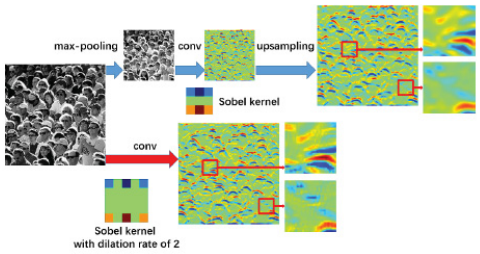

Figura 10. Representação da camada de Convolução Dilatada
Fonte: Yuhong [21] (2018)

Como etapa de pré-processamento para a implementação da CSRnet, aplicou-se a técnica de escala cinza nas imagens provenientes nos *frames*. No *output* dos mapas de densidade, aplicou-se um filtro gaussiano para suavizar o mapa para reduzir ruído ou variações bruscas - uma abordagem similar foi utilizada em [22].

1) Inicializa o modelo já treinado de VGG16 e suas configurações;
2) Inicializa vídeo a ser analisado;
3) Aplica o pré-processamento da imagem do *frame*;
4) Define as configurações da transformação de pré-processamento: Converte frame para PIL, faz o redimensionamento, converte para tensor e normaliza com média e desvio padrão;
5) Para cada *frame*: pré-processa o *frame* e o transforma em tensor antes de envia-lo ao modelo. O CSRNet, por sua vez, retorna um mapa de densidade , onde cada pixel representa a probabilidade de presença de uma pessoa. A soma dos pixels do mapa nos retorna a contagem das pessoas no *frame* em questão.

Considerando os classificadores de nível de densidade e a camada de dilatação, tem-se uma solução de fácil treinamento e com melhor desempenho em métricas de contagem e qualidade de mapas de densidade (*MAE*, *MSE*, *PSNR* e *SSIM*) - demonstradas nas análises em cima dos *datasets* ShanghaiTech, UCF CC 50, WorldExpo'10 e TRANCOS [21]. Sendo assim, CSRNet é uma solução mais enxuta, precisa e escalável, consolidando-se como uma referência para análise de cenas congestionadas.

C. Tecnologias

Todo o trabalho de geração da base de dados de imagens provenientes do *streaming* e dos algoritmos de identificação de humanos foi desenvolvido em um notebook da marca Dell de modelo Latitude 5440 com processador de 13ª geração Intel

Core i5-1345U com 12 CPUs em uma frequencia máxima de 1.6 GHz por núcleo. No Python, foram utilizadas as seguintes bibliotecas principais:

- Linguagem Python, versão 3.8.5;
- VS Code, versão 2025.1.100;
- Biblioteca *cvzone*, versão 1.5.6;
- Biblioteca *filterpy*, versão 1.4.5;
- Biblioteca *google-auth*, versão 2.39.0;
- Biblioteca *imageio*, versão 2.35.1;
- Biblioteca *matplotlib*, versão 3.7.5;
- Biblioteca *mpmath*, versão 1.3.0;
- Biblioteca *numpy*, versão 1.24.4;
- Biblioteca *opencv-python*, versão 4.5.4.60;
- Biblioteca *pandas*, versão 2.0.3;
- Biblioteca *pip*, versão 20.1.1;
- Biblioteca *scikit-image*, versão 0.19.3;
- Biblioteca *scipy*, versão 1.10.1;
- Biblioteca *seaborn*, versão 0.13.2;
- Biblioteca *tensorboard*, versão 2.14.0;
- Biblioteca *torchvision*, versão 0.19.1;
- Biblioteca *ultralytics*, versão 8.0.26

II. RESULTADOS E DISCUSSÕES

A seguir são apresentados os resultados gerados a partir da aplicação de cada um dos quatro modelos de visão computacional implementados neste relatório em cinco vídeos de multidões, com o objetivo de contar de forma assertiva a quantidade de pessoas por *frame*. Com o intuito de avaliar o caráter generalista de cada modelo, não houve alterações nos métodos de pré-processamento, apenas nos valores de parâmeros dentro de cada modelo ao percorrer por cada um dos vídeos. Considerando 3 cenários de parâmetros distintos para cada um dos 4 modelos em 5 vídeos diferentes, tem-se um total de 60 resultados a serem analisados neste seção.

Toda a mensuração foi gerada através de uma comparação com o *Ground Truth*, que consiste no conjunto de dados rotulados manualmente por humanos e usado como referência para validar ou treinar modelos de visão computacional. No contexto de contagem de pessoas, o *ground truth* refere-se ao número real de pessoas presentes em cada *frame* de um vídeo. Este processo permite comparar os resultados das métricas dos modelos deste relatório.

Considerando o caráter generalista adotado neste estudo de cada um dos modelos, na Tabela I tem-se o detalhamento da etapa de pré-processamento. O filtro Gaussiano do modelo CSRnet foi apontado com um asterisco pois foi utilizado em apenas um dos três cenários do modelo.

Tabela I
ALGORITMOS, PRÉ-PROCESSAMENTO E MODELOS PRÉ-TREINADOS UTILIZADOS.

ALGORITMO	PRÉ-PROCESSAMENTO	MODELO PRÉ-TREINADO
HAAR Cascade	Escala Cinza, Máscara e Filtro de Medianas	haarcascade_upperbody.xml
HOG + SVM	Escala Cinza, Máscara e CLAHE	SVMDetector do cv2.HOGDescriptor
YOLO	Escala Cinza, Máscara e CLAHE	yolov8m.pt
CSRnet	Escala Cinza, Máscara e Filtro Gaussiano*	PartAmodel_best.pth (Modelo VGG16)

Fonte: O autor (2025).

Na sequência, são apontadas as Tabelas II, III, IV e V responsáveis pelos parâmetros dos algoritmos considerando os três cenários distintos para cada modelo. A Tabela II detalha os parâmetros utilizados nos três cenários do *Haar Cascade*: sensível, intermediário e conservador. A Tabela III trás os mesmos cenários para o modelo HOG. A Tabela IV detalha as diferentes variáveis para os cenários do *baseline*, conservador e *tuned* do YOLO e por fim a Tabela V apresente os parâmetros do CSRnet considerando os cenários *baseline*, *resized* e *custom.*

Tabela II
ALGORITMO, CENÁRIO E PARÂMETROS - HAAR CASCADE

ALGORITMO	CENÁRIO	PARÂMETROS
HAAR Cascade	Cenário Sensível	*scalefactor*: 1.01, *minneighbors*: 3 e *minsize*: (20, 20)
HAAR Cascade	Cenário Intermediário	*scalefactor*: 1.03, *minneighbors*: 4 e *minsize*: (30, 30)
HAAR Cascade	Cenário Conservador	*scalefactor*: 1.05, *minneighbors*: 8 e *minsize*: (40, 40)

Fonte: O autor (2025).

Tabela III
ALGORITMO, CENÁRIO E PARÂMETROS - HOG

ALGORITMO	CENÁRIO	PARÂMETROS
HOG	Cenário Sensível	*winstride*: (4,4), *padding*: (8,8), *scale*: 1.01 e *hitthreshold*: 0.0
HOG	Cenário Intermediário	*winstride*: (6,6), *padding*: (12,12), *scale*: 1.03 e *hitthreshold*: 0.2
HOG	Cenário Conservador	*winstride*: (8,8), *padding*: (16,16), *scale*: 1.05 e *hitthreshold*: 0.5

Fonte: O autor (2025).

Tabela IV
ALGORITMO, CENÁRIO E PARÂMETROS - YOLO

ALGORITMO	CENÁRIO	PARÂMETROS
YOLO	Cenário Baseline	*conf_threshold*: 0,3, *iou_threshold_yolo*: 0.3, *t_max_age*: 10, *t_min_hits*: 2, *t_iou_threshold*: 0.1
YOLO	Cenário Conservador	*conf_threshold*: 0,5, *iou_threshold_yolo*: 0.3, *t_max_age*: 10, *t_min_hits*: 2, *t_iou_threshold*: 0.1
YOLO	Cenário Tuned	*conf_threshold*: 0,3, *iou_threshold_yolo*: 0.5, *t_max_age*: 5, *t_min_hits*: 3, *t_iou_threshold*: 0.2

Fonte: O autor (2025).

Tabela V
ALGORITMO, CENÁRIO E PARÂMETROS - CSRNET

ALGORITMO	CENÁRIO	PARÂMETROS
CSRnet	Cenário Baseline	*resized_shape*: (768, 1024), *use_gaussian*: False, *sigma_value*: 0, *normalize_params*: *mean*: [0.485, 0.456, 0.406], *std*: [0.229, 0.224, 0.225]
CSRnet	Cenário Resized	*resized_shape*: (640, 960), *use_gaussian*: False, *sigma_value*: 0, *normalize_params*: *mean*: [0.485, 0.456, 0.406], *std*: [0.229, 0.224, 0.225]
CSRnet	Cenário Custom	*resized_shape*: (768, 1024), *use_gaussian*: True, *sigma_value*: 1.0, *normalize_params*: *mean*: [0.5, 0.5, 0.5], *std*: [0.5, 0.5, 0.5]

Fonte: O autor (2025).

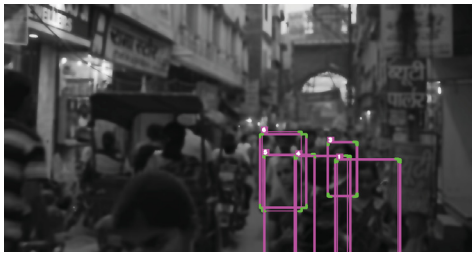

Figura 11. Exemplo de identificação em *frame* do VD1: Modelo YOLO v8
Fonte: O autor (2025)

A. Medida de qualidade dos modelos

A assertividade dos modelos deste relatório foi mensurada a partir das métricas muito presentes na avaliação de modelos regressivos: *Root Mean Squared Error* (*RMSE*) e *Mean Absolute Error* (*MAE*), abordadas no trabalho de redes neurais convolucionais dilatadas, ou *CSRNet*, de Yuhong [21].

Considerando os 60 cenários de resultados, executando 3 configurações distintas de parâmetros para 4 modelos em 5 vídeos, o algoritmo com melhor *trade-off* entre processamento e performance de execução é o *YOLO modelo_tuned*, com as médias de *MAE* = 0,867, *MSE* = 1,564 e *RMSE* = 1,173. Para cada tabela contendo os desempenhos dos modelos por vídeo, serão destacados em negrito o cenário de parâmetros com menor erro absoluto nas detecções.

A Tabela VI apresenta o resultado das três métricas obtidas para cada um dos quatro modelos ao aplica-los no vídeo VD1, considerando três cenários de parâmetros para cada um. Os quatro melhores cenários que representam uma boa assertividade na contagem de pessoas por *frame* são, respectivamente, *YOLO modelo_baseline* com *MAE* = 1,183, *MSE* = 2,093 e *RMSE* = 1,447. *YOLO modelo_tuned* com *MAE* = 1,253, *MSE* = 2,335 e *RMSE* = 1,528. *CSRnet modelo_resized* com *MAE* = 1,304, *MSE* = 3,156 e *RMSE* = 1,776 e por fim *CSRnet modelo_baseline* com *MAE* = 1,311, *MSE* = 3,241 e *RMSE* = 1,800. Em termos de tempo de processamento, destaca-se o *modelo_resized* da CSRnet que obteve 250 segundos de tempo total, sendo o dobro da performance do modelo de menor *MAE*, *YOLO modelo_baseline*.

Tabela VI
COMPARAÇÃO DE DESEMPENHO DOS MODELOS NO VÍDEO VD1.

Algortimo	Cenário	MAE	MSE	RMSE	Processamento (s)
HAAR	modelo_sensivel	1,969	5,759	2,4	56
HAAR	modelo_intermediario	4,23	18,805	4,337	77
HAAR	modelo_conservador	4,693	22,724	4,767	35
HOG	modelo_sensivel	3,689	15,206	3,9	427
HOG	modelo_intermediario	3,868	16,35	4,044	714
HOG	modelo_conservador	4,654	22,568	4,751	140
YOLO	**modelo_baseline**	**1,183**	**2,093**	**1,447**	**540**
YOLO	modelo_conservador	2,082	5,342	2,311	157
YOLO	modelo_tuned	1,253	2,335	1,528	923
CSRnet	modelo_baseline	1,311	3,241	1,8	972,5
CSRnet	modelo_resize	1,304	3,156	1,776	250
CSRnet	modelo_gaussiano	3,323	12,56	3,544	1695

Fonte: O autor (2025)

Para o segundo vídeo, tem-se na Tabela VII os resultado das três métricas obtidas para cada um dos quatro modelos considerando as três configurações de parâmetro. Neste caso, os resultados assertivos de contagem ficaram com os três modelos *YOLO*: *modelo_baseline* com *MAE* = 0,453, *MSE* = 0,636 e *RMSE* = 0,798, *modelo_tuned* com *MAE* = 0,484, *MSE* = 0,708 e *RMSE* = 0,841 e *modelo_conservador* com *MAE* = 0,777, *MSE* = 1,441 e *RMSE* = 1,201. Embora para o segundo vídeo tenha-se tempo de processamento superior nos modelos *HAAR* e *HOG* (respectivamente, 673 e 645 segundos), seus altos valores para as métricas de erro na contagem excluem sua consideração na análise de viabilidade do modelo.

Tabela VII
COMPARAÇÃO DE DESEMPENHO DOS MODELOS NO VÍDEO VD2.

Algortimo	Cenário	MAE	MSE	RMSE	Processamento (s)
HAAR	modelo_sensivel	6,381	45,888	6,774	752
HAAR	modelo_intermediario	1,754	4,648	2,156	673
HAAR	modelo_conservador	3,284	12,728	3,568	698
HOG	modelo_sensivel	1,9	5,304	2,303	645
HOG	modelo_intermediario	2,1	6,375	2,525	1102
HOG	modelo_conservador	3,16	11,074	3,328	188
YOLO	**modelo_baseline**	**0,453**	**0,636**	**0,798**	**1202**
YOLO	modelo_conservador	0,777	1,441	1,201	1176
YOLO	modelo_tuned	0,484	0,708	0,841	1228
CSRnet	modelo_baseline	1,367	2,622	1,619	2995,5
CSRnet	modelo_resize	1,235	2,289	1,513	1021
CSRnet	modelo_gaussiano	2,295	5,797	2,408	4970

Fonte: O autor (2025)

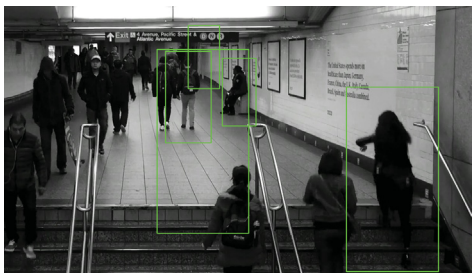

Figura 12. Exemplo de identificação em *frame* do VD2: Modelo HOG
Fonte: O autor (2025)

A Tabela VIII apresenta os resultados para todos os cenários de modelos aplicados ao VD3. Desta vez, dois modelos de *CSRnet* obtiveram melhores métricas: *modelo_resize* com *MAE* = 0,680, *MSE* = 0,804 e *RMSE* = 0,896 e o *modelo_baseline* com *MAE* = 0,713, *MSE* = 0,789 e *RMSE* = 0,888. Na sequência da *CSRnet*, os modelos mais assertivos foram os cenários

gerados a partir do *YOLO*: *modelo_tuned* com *MAE* = 0,736, *MSE* = 0,941 e *RMSE* = 0,970, *modelo_baseline* com *MAE* = 0,991, *MSE* = 1,249 e *RMSE* = 1,118 e *modelo_conservador* com *MAE* = 1,129, *MSE* = 1,716 e *RMSE* = 1,310. Torna-se necessário apontar que o melhor modelo *CSRnet* levou quase 5 vezes o tempo de processamento do melhor modelo *YOLO*, sendo um critério relevante na definição do melhor modelo para o terceiro vídeo.

Tabela VIII
COMPARAÇÃO DE DESEMPENHO DOS MODELOS NO VÍDEO VD3.

Algortimo	Cenário	MAE	MSE	RMSE	Processamento (s)
HAAR	modelo_sensivel	2,548	7,751	2,784	974,5
HAAR	modelo_intermediario	4,273	18,701	4,324	1035
HAAR	modelo_conservador	4,557	21,085	4,592	914
HOG	modelo_sensivel	1,238	2,223	1,491	752
HOG	modelo_intermediario	2,085	4,965	2,228	720
HOG	modelo_conservador	4,015	17,111	4,137	784
YOLO	modelo_baseline	0,991	1,249	1,118	131
YOLO	modelo_conservador	1,129	1,716	1,310	87
YOLO	modelo_tuned	0,736	0,941	0,97	175
CSRnet	modelo_baseline	0,713	0,789	0,888	857
CSRnet	**modelo_resize**	**0,680**	**0,804**	**0,896**	**762**
CSRnet	modelo_gaussiano	5,050	26,270	5,125	952

Fonte: O autor (2025)

Figura 13. Exemplo de identificação em *frame* do VD3: Modelo *Haar Cascade*
Fonte: O autor (2025)

Pode-se observar os resultados para a quarta amostra de vídeo na Tabela IX. Similar à amostra VD2, para a VD4 tem-se que os resultados mais assertivos em termos de contagem ficaram com os três modelos *YOLO*: *modelo_tuned* com *MAE* = 0,423, *MSE* = 0,477 e *RMSE* = 0,691, *modelo_conservador* com *MAE* = 0,432, *MSE* = 0,523 e *RMSE* = 0,723 e *modelo_baseline* com *MAE* = 0,468, *MSE* = 0,553 e *RMSE* = 0,744. Na sequência, tem-se o modelo *CSRNET modelo_resize* com *MAE* = 0,843, *MSE* = 1,181 e *RMSE* = 1,087 e pela primeira vez no ranqueamento de melhores resultados o *HOG modelo_intermediario* com *MAE* = 0,879, *MSE* = 1,622 e *RMSE* = 1,274. Em termos de performance de execução, embora tenham uma melhor assertividade nas contagens, os três modelos *YOLO* possuem quase o dobro de tempo de processamento que as soluções *CSRnet* e *HOG*.

Tabela IX
COMPARAÇÃO DE DESEMPENHO DOS MODELOS NO VÍDEO VD4.

Algortimo	Cenário	MAE	MSE	RMSE	Processamento (s)
HAAR	modelo_sensivel	1,196	2,429	1,559	56
HAAR	modelo_intermediario	1,498	3,438	1,854	71
HAAR	modelo_conservador	2,033	5,538	2,353	41
HOG	modelo_sensivel	1,495	3,906	1,976	36,5
HOG	modelo_intermediario	0,879	1,622	1,274	47
HOG	modelo_conservador	0,94	1,84	1,356	26
YOLO	modelo_baseline	0,468	0,553	0,744	80,5
YOLO	modelo_conservador	0,432	0,523	0,723	82
YOLO	**modelo_tuned**	**0,423**	**0,477**	**0,691**	**79**
CSRnet	modelo_baseline	1,097	1,749	1,323	63
CSRnet	modelo_resize	0,843	1,181	1,087	42
CSRnet	modelo_gaussiano	3,804	15,459	3,932	84

Fonte: O autor (2025)

Figura 14. Exemplo de identificação em *frame* do VD4: Modelo Yolo v8
Fonte: O autor (2025)

A Tabela X apresenta os resultados para todos os cenários de modelos aplicados ao quinto vídeo, ou VD5. Assim como os resultados obtidos para o terceiro vídeo, os dois modelos de *CSRnet* obtiveram melhores métricas: *modelo_baseline* com *MAE* = 0,898, *MSE* = 1,296 e *RMSE* = 1,138 e o *modelo_resize* com *MAE* = 0,978, *MSE* = 1,485 e *RMSE* = 1,219. Apesar dos três modelos *YOLO* estarem na sequência do ranqueamento de assertividade, o tempo de processamento médio para estes casos é 23% maior que as duas soluções *CSRnet*, não sendo o melhor modelo no *trade-off* assertividade e *performance* para o VD5.

Tabela X
COMPARAÇÃO DE DESEMPENHO DOS MODELOS NO VÍDEO VD5.

Algortimo	Cenário	MAE	MSE	RMSE	Processamento (s)
HAAR	modelo_sensivel	4,455	21,848	4,674	103,5
HAAR	modelo_intermediario	5,823	35,978	5,998	105
HAAR	modelo_conservador	6,391	42,848	6,546	102
HOG	modelo_sensivel	2,935	12,741	3,569	106,5
HOG	modelo_intermediario	3,336	13,132	3,624	107
HOG	modelo_conservador	4,617	24,95	4,995	106
YOLO	modelo_baseline	1,502	3,766	1,941	127,5
YOLO	modelo_conservador	1,781	4,94	2,223	128
YOLO	modelo_tuned	1,438	3,358	1,833	127
CSRnet	**modelo_baseline**	**0,898**	**1,296**	**1,138**	**104,5**
CSRnet	modelo_resize	0,978	1,485	1,219	102
CSRnet	modelo_gaussiano	4,968	25,913	5,09	107

Fonte: O autor (2025)

Na Figura 15 temos uma representação dos mapas de densidade do CSRnet, aplicada ao VD5: Diferente dos modelos de *Haar Cascade*, HOG ou YOLO que delimitam regiões

de *bounding boxes* em cima dos objetos, a CSRnet - já treinada com exemplos de imagens contendo anotações dos pontos centrais das cabeças das pessoas na imagem e seus respectivos *ground truths* - gera mapas de densidade que têm picos localizados exatamente onde estão as pessoas. Estes picos por sua vez são suavizados por um filtro gaussiano que cria o efeito contínuo do mapa de calor representado na imagem.

Figura 15. Exemplo de identificação em *frame* do VD5: Modelo CSRnet
Fonte: O autor (2025)

B. Discussões

Os resultados com as médias de métricas por algoritmo e configuração está representado na Tabela XI. O modelo generalista com menor média de erros é o *YOLO modelo_tuned*, com as médias de *MAE* = 0,867, *MSE* = 1,564 e *RMSE* = 1,173.

Tabela XI
RANQUEAMENTO DAS MÉDIAS FINAIS DE MAE, MSE E RSME DOS MODELOS, POR CENÁRIO.

Algortimo	Cenário	MAE	MSE	RMSE	Processamento (s)
YOLO	modelo_tuned	0,867	1,564	1,173	506,4
YOLO	modelo_baseline	0,920	1,660	1,209	416,2
CSRnet	modelo_resize	1,008	1,783	1,298	435,4
CSRnet	modelo_baseline	1,077	1,939	1,354	998,5
YOLO	modelo_conservador	1,240	2,792	1,553	326
HOG	modelo_sensivel	2,251	7,876	2,648	393,4
HOG	modelo_intermediario	2,454	8,489	2,739	538
HAAR	modelo_sensivel	3,310	16,735	3,638	388,4
HOG	modelo_conservador	3,477	15,509	3,713	248,8
HAAR	modelo_intermediario	3,516	16,314	3,734	392,2
CSRnet	modelo_gaussiano	3,888	17,200	4,020	1561,6
HAAR	modelo_conservador	4,191	20,985	4,365	358

Fonte: O autor (2025)

Apesar do domínio dos algortimos *YOLO* e *CSRnet*, é válido entender as possíveis causas da baixa assertividade dos métodos tradicionais: De acordo com Costa [23], ambos os métodos de extração de características LBP e HOG são sensíveis à ruídos na imagem, podendo inviabilizar a posterior classificação de um SVM do próprio *pipeline*. Neste relatório, como propositalmente foram congeladas as etapas de pré-processamento para avaliar o caráter generalista dos modelos, acabou-se por inviabilizar uma boa acurácia de contagem em métodos tradicionais de visão computacional.

No âmbito dos algoritmos CSRnet foi testado dentre os três cenários propostos o *trade-off* entre assertividade na contagem e performance do modelo, visto que as redes dilatadas possuem um maior tempo de processamento [24]. Como algoritmo otimizado, o cenário *modelo_resized* foi configurado para uma resolução de entrada menor de (640, 960). Esta modificação permitiu um tempo de processamento 2.3 vezes menos do que o *modelo_baseline* e 3.5 vezes menor que o *modelo_gaussiano*. Na Figura 16 pode-se observar que, apesar da redução na resolução de entrada, o *modelo_resized* apresentou as melhores métricas médias dentre os algortimos da CSRnet, com *MAE* = 1,008 e tempo de processamento médio de 435 segundos.

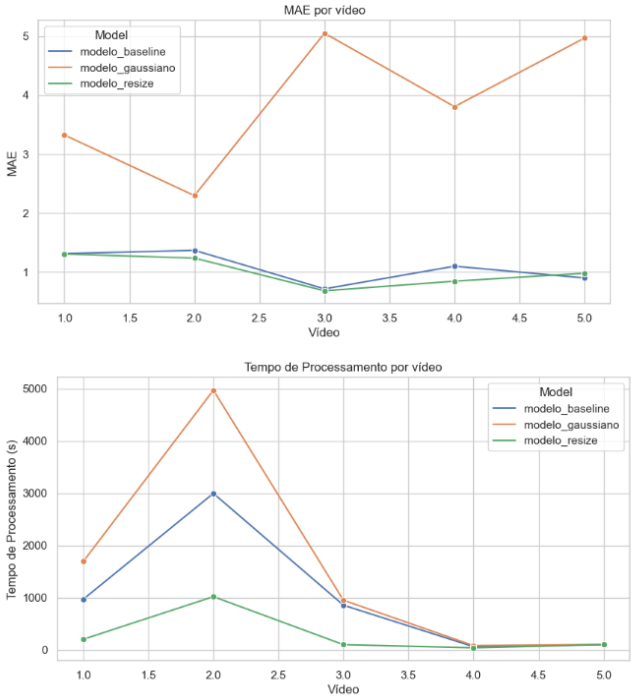

Figura 16. Comparativo das métricas e tempo de performance para as diferentes versões do CSRnet
Fonte: O autor (2025)

Outro teste realizado nos algoritmos CSRnet foi a implantação de um filtro Gaussiano no *output* do *modelo_gaussiano*. Apesar das melhores reportadas no estudo de Yuhong [21] nas métricas de detecção, para os resultados do deste relatório o filtro gaussiano piorou a identificação das pessoas a partir dos *frames*, deixando o *modelo_gaussiano* atrás no ranqueamento de assertividade até dos *modelos_sensiveis* e *modelos_conservadores* do HOG.

Para os cenários do *YOLO*, foram-se testadas configurações com base na literatura voltadas para dois cenários: confiabilidade da detecção através dos parâmetros do próprio *YOLO* e através dos parâmetros do *tracking*. Um aumento no *threshold* de confiança faz o algoritmo exigir maior certeza para considerar uma detecção válida, reduzindo falsos positivos [17]. Além do *threshold* de confiança, pode-se garantir a qualidade de detecção com incremento do IoU - quanto mais alto mais ele garante que apenas detecções com alta sobreposição sejam consideradas, evitando contagens duplicadas [25]. Com base nestes

dois artigos foi-se gerado o cenário *YOLO modelo_conservador* que, apesar de possuir métricas piores que os outros modelos *YOLO* implementados, tem um tempo de processamento menor, chegando a ser até 55% mais rápido do que o cenário *YOLO modelo_tuned*. Considerando os parâmetros de *tracking*, pela literatura tem-se que ao trabalhar com valores maiores de *max_age*, *min_hits* e um valor equilibrado de *iou_threshold* a consistência de identificação de objetos é garantida ao passo que suprime associações errôneas, como identificar uma sombra ou quadro na parede como pessoa [26]. Ao comparar o *modelo_baseline* com a otimização das configurações de *tracking* previstos no *modelo_tuned*, tem-se uma melhora média de 6% nas métricas de acurária de contagem, ao passo que o tempo de processamento aumenta em 15%.

Ao considerar-se o objetivo generalista deste relatório aliado às métricas de assertividade na contagem e tempo de processamento, a decisão mais assertiva é de seguir com o *YOLO modelo_tuned*, que possui as menores médias de *MAE*, *MSE* e *RMSE* - respectivamente: 0,867, 1,564 e 1,173 - ao passo que mantém um tempo de processamento relativamente próximo às médias do método *Haar Cascade* que, dos quatro modelos, foi o mais performático para realizar as contagens.

C. Trabalhos Futuros

Este relatório concluiu um estudo de caso que aponta dentre algumas soluções de contagem de pessoas em vídeos qual o método de *machine learning* mais promissor, em termos de assertividade e *performance*. A partir do entendimento do potencial do YOLO e da obtenção de dados de gravações de linhas de produção reais, serão desdobradas duas frentes de trabalho: a primeira com foco em entender o potencial de melhora nos resultados a partir de tratamentos e metodologias de pré-processamento de imagens - algo congelado neste relatório - e a segunda frente voltada à não utilização de modelos pré-treinados, requisitando trabalho adicional na aquisição de imagens, aplicação de *labels* e posterior treinamento e validação de um modelo *self-made*.

REFERÊNCIAS

[1] D. H. Ballard and C. M. Brown, *Computer Vision*. Prentice Hall Professional Technical Reference, 1st ed., 1982.

[2] A. Y.-T. Ng, "Neural networks deep learning specialization - online course." Disponível em: <https://www.coursera.org/learn/neural-networks-deep-learning?specialization=deep-learning>. Acesso em: 06 de mar. de 2025.

[3] F. Barelli, *Introdução à Visão Computacional - Uma abordagem prática com Python e OpenCV*. São Paulo: Casa do Código, 1 ed., 2019.

[4] C. Steger, M. Ulrich, and C. Wiedemann, *Machine Vision Algorithms and Applications*. Wiley VCH, 01 2018.

[5] L. C. Moitinho and A. Benicasa, "Aprendizado de máquina para o auxílio à localização de pessoas em ambientes indoor monitorados por câmeras," in *Anais Estendidos do XIX Simpósio Brasileiro de Sistemas de Informação*, (Porto Alegre, RS, Brasil), pp. 71–80, SBC, 2023.

[6] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. USA: Prentice-Hall, Inc., 2006.

[7] B. F. Araújo, "Sistema de visão de máquina para detecção e localização automática de peças utilizando raspberry pi," Master's thesis, Instituto Federal de Paraíba - Campus João Pessoa, 2019.

[8] "Banco de dados de vídeos." https://www.pexels.com/, 2025. Acesso em: 25 mai. 2025.

[9] H. Yun and D. Park, "Efficient object detection based on masking semantic segmentation region for lightweight embedded processors," *Sensors 2022*, vol. 22, 2022.

[10] M. Yousif, "Enhancing the accuracy of image classification using deep learning and preprocessing methods," *Artificial Intelligence Robotics Development Journal*, 01 2024.

[11] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–I, 2001.

[12] R. Aras and H. Endang, "Implementation of haar cascade and adaboost algorithms in photo classification on social networks," *Inspiration: Jurnal Teknologi Informasi dan Komunikasi*, vol. 13, pp. 59–68, 06 2023.

[13] B. P. Le, H. Jeon, N. Truong, and J. Hak, "Applying the haar-cascade algorithm for detecting safety equipment in safety management systems for multiple working environments," *Electronics*, vol. 8, p. 1079, 09 2019.

[14] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1491–1498, 2006.

[15] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[16] A. Géron, *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn TensorFlow*. Alta Books, 2019.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[18] S. Luetto, "People counting using detection networks and self calibrating cameras on edge computing," Master's thesis, Politecnico di Torino, 2019.

[19] H. Wang, "Detection of humans in video streams using convolutional neural networks," Master's thesis, KTH Royal Institute of Technology, 2017.

[20] R.-C. Chen, C. Dewi, Y.-C. Zhuang, and J.-K. Chen, "Contrast limited adaptive histogram equalization for recognizing road marking at night based on yolo models," *IEEE Access*, vol. 11, pp. 92926–92942, 2023.

[21] Y. Li, X. Zhang, and D. Chen, "Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1091–1100, 2018.

[22] M. M. Oghaz, A. R. Khadka, V. Argyriou, and P. Remagnino, "Content-aware density map for crowd counting and density estimation," 2019.

[23] G. B. P. da Costa, W. A. Contato, T. S. Nazare, J. E. S. B. Neto, and M. Ponti, "An empirical study on the effects of different types of noise in image classification tasks," 2016.

[24] Y. Hou, C. Li, Y. Lu, L. Zhu, Y. Li, H. Jia, and X. Xie, "Enhancing and dissecting crowd counting by synthetic data," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2539–2543, 2022.

[25] L. Nguyen, Y. Son, and T. M. Dat, "Empirical evaluation and analysis of yolo models in smart transportation," 09 2024.

[26] X. Xiao and X. Feng, "Multi-object pedestrian tracking using improved yolov8 and oc-sort," *Sensors*, vol. 23, p. 8439, 10 2023.