**UNIVERSIDADE FEDERAL DE JUIZ DE FORA**
**PROGRAMA DE PÓS GRADUAÇÃO EM MODELAGEM COMPUTACIONAL**

**Luan Patrick Silva Feitosa de Melo**

# Identificação, rastreamento e contagem de suínos em tempo real através de imagens

Juiz de Fora

2025

**Luan Patrick Silva Feitosa de Melo**

# Identificação, rastreamento e contagem de suínos em tempo real através de imagens

Dissertação apresentada ao Programa de Pós Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Modelagem Computacional. Área de concentração: Modelagem Computacional

Orientador: Profa. Dra. Priscila Vanessa Zabala Capriles Goliatt

Coorientador: Prof. Dr. Leonardo Goliatt da Fonseca

Juiz de Fora

2025

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Melo, Luan Patrick Silva Feitosa de.
Identificação, rastreamento e contagem de suínos em tempo real através de imagens / Luan Patrick Silva Feitosa de Melo. – 2025.
77 f. : il.

Orientador: Priscila Vanessa Zabala Capriles Goliatt
Coorientador: Leonardo Goliatt da Fonseca
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, . Programa de Pós Graduação em Modelagem Computacional, 2025.

1. Suinocultura. 2. Visão Computacional. 3. Rastreamento de Objetos. I. Capriles, Priscila, orient. II. Goliatt, Leonardo, coorient. III. Título.

**Luan Patrick Silva Feitosa de Melo**

**Identificação, rastreamento e contagem de suínos em tempo real através de imagens**

Dissertação apresentada ao Nome do Curso ou Programa da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em (*). Área de concentração:

Aprovada em 11 de abril de 2025.

BANCA EXAMINADORA

**Prof.ª Dr.ª. Priscila Vanessa Zabala Capriles Goliatt -** Orientadora
Universidade Federal de Juiz de Fora

**Prof. Dr. Leonardo Goliatt da Fonseca**
Universidade Federal de Juiz de Fora

**Prof.ª Dr.ª. Luciana Conceição Dias Campos**
Universidade Federal de Juiz de Fora

**Prof. Dr. Renato Ribeiro Aleixo**
Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

**João Victor Pinto Coelho**
Grupo Cabo Verde

Juiz de Fora, 09/04/2025.

Documento assinado eletronicamente por **Priscila Vanessa Zabala Capriles Goliatt**, **Professor(a)**, em 26/04/2025, às 10:15, 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 **RENATO RIBEIRO ALEIXO**, **Usuário Externo**, em 29/04/2025, às 19:00, 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 **João Victor Pinto Coelho**, **Usuário Externo**, em 10/06/2025, às 14:54, 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 **Leonardo Goliatt da Fonseca**, **Professor(a)**, em 10/06/2025, às 19:45, 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 **2343036** e o código CRC **B8CB3658**.

## AGRADECIMENTOS

Aos meus pais, Luiz Carlos e Edna, pelo amor incondicional, pelo exemplo de dedicação e pelo apoio constante em todos os momentos da minha vida. E a toda a minha família por sempre me apoiar.

À minha esposa Lorraine, pelo companheirismo, compreensão e incentivo diário que foram fundamentais durante toda a jornada deste trabalho.

Ao Guilherme Amadeu, pelo suporte técnico essencial na implementação da conexão remota na granja, cuja colaboração foi determinante para o desenvolvimento deste projeto.

À minha equipe no iFood Pago, em especial ao Jackson Garcia, pela liderança inspiradora, e ao Leandro Villela, pelo apoio inestimável nas situações complicadas que enfrentei em código — ambos fundamentais para superar desafios técnicos ao longo deste trabalho.

Aos meus orientadores, Priscila Capriles e Leonardo Goliatt, pela orientação precisa, pelas valiosas contribuições acadêmicas e pela confiança depositada em mim ao longo desta pesquisa.

Ao grupo Cabo Verde, em especial ao João Victor e à dedicada equipe da granja, pela abertura e total suporte durante a fase prática deste trabalho, que possibilitaram a realização dos experimentos em condições reais de campo.

A todos vocês, minha mais sincera gratidão.

“Aqueles que são loucos o suficiente para acharem que podem mudar o mundo são os que de fato mudam”.
**Steve Jobs**

**RESUMO**

O presente trabalho propõe o desenvolvimento de um sistema embarcado para identificação, rastreamento e contagem automática de suínos em tempo real, com o objetivo de minimizar erros no processo de embarque, etapa crítica da cadeia produtiva da suinocultura. Utilizando a plataforma NVIDIA Jetson Nano e o algoritmo de detecção YOLOv12 em suas variações de modelo, associados aos rastreadores SORT e ByteTrack, foi possível realizar uma análise comparativa detalhada quanto à acurácia, tempo de inferência e estabilidade dos identificadores gerados. O sistema foi avaliado em ambiente real de produção, com vídeos capturados em granja comercial, totalizando seis cenários distintos e mais de mil imagens anotadas. Os resultados demonstram que a combinação do modelo YOLOv12-nano com o rastreador ByteTrack alcançou o melhor desempenho, apresentando alta precisão na contagem e robustez frente a desafios como oclusões, sobreposição de animais e variações de iluminação. A pesquisa contribui para a modernização da suinocultura brasileira, oferecendo uma solução de baixo custo, eficiente e viável para a automação do controle de fluxo de animais, impactando positivamente a gestão zootécnica e logística da produção.

Palavras-chave: visão computacional; contagem automática; suinocultura; Jetson Nano; YOLOv12; ByteTrack; SORT.

**ABSTRACT**

This study proposes the development of an embedded system for real-time identification, tracking, and automatic counting of pigs, aiming to minimize errors during the loading process—a critical stage in the pig production chain. Using the NVIDIA Jetson Nano platform and the YOLOv12 detection algorithm across its model variations, combined with the SORT and ByteTrack trackers, a comprehensive comparative analysis was conducted focusing on accuracy, inference time, and identifier stability. The system was evaluated in real-world production conditions, with videos captured at a commercial farm, totaling six distinct scenarios and over one thousand annotated images. Results showed that the YOLOv12-nano model, paired with the ByteTrack tracker, achieved the best performance, delivering high counting accuracy and robustness against challenges such as occlusions, animal overlap, and lighting variations. This research contributes to the modernization of Brazilian pig farming by offering a low-cost, efficient, and practical solution for automating animal flow control, positively impacting livestock management and production logistics.

Keywords: computer vision; automatic counting; pig farming; Jetson Nano; YOLOv12; ByteTrack; SORT.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AI | Inteligência Artificial (Artificial Intelligence) |
| AMP | Precisão Mista Automática (Automatic Mixed Precision) |
| AP | Precisão Média (Average Precision) |
| CNN | Rede Neuronal Convolucional (Convolutional Neural Network) |
| CBAM | Módulo de Atenção de Bloco Convolucional (Convolutional Block Attention Module) |
| FPS | Quadros por Segundo (Frames Per Second) |
| FPN | Rede Piramidal de Características (Feature Pyramid Network) |
| GPU | Unidade de Processamento Gráfico (Graphics Processing Unit) |
| HRST | Rede de Resolução Alta com Transformer (High-Resolution Swin Transformer) |
| HRNet | Rede de Alta Resolução (High-Resolution Network) |
| ID | Identificador Único |
| IoU | Interseção sobre União (Intersection over Union) |
| mAP | Média de Precisão (mean Average Precision) |
| MLP | Perceptron Multicamadas (Multi-Layer Perceptron) |
| MOTA | Multi-Object Tracking Accuracy |
| MOTP | Multiple Object Tracking Precision |
| MS COCO | Microsoft Common Objects in Context |
| NMS | Supressão Não-Máxima (Non-Maximum Suppression) |
| OCDE | Organização para a Cooperação e Desenvolvimento Econômico |
| R-CNN | Region-based Convolutional Neural Networks |
| ResNet | Residual Neural Network |
| SORT | Simple Online and Realtime Tracking |
| YOLO | You Only Look Once |

## SUMÁRIO

# 1 INTRODUÇÃO

A suinocultura brasileira tem se modernizado e expandido em termos tecnológicos e comerciais, consolidando o país como o terceiro maior produtor e o quinto maior exportador mundial de carne suína. Esse cenário exige processos cada vez mais precisos e eficientes em todas as etapas produtivas, especialmente no embarque dos animais – momento crítico no qual erros de contagem podem gerar prejuízos significativos aos produtores e às empresas. Dados recentes indicam que aproximadamente 80,6% dos estabelecimentos suinícolas no Brasil são geridos pela agricultura familiar. Entretanto, grande parte da produção está concentrada em granjas de médio e grande porte, especialmente nas regiões Sul e Sudeste, onde predominam sistemas integrados e intensivos, com alta tecnificação e volume expressivo de animais comercializados (1).

Apesar da expansão tecnológica e da posição de destaque, as projeções para 2025 indicam um crescimento moderado na produção e exportação de carne suína, mesmo diante de um cenário global desafiador. Os dados sugerem que, embora o setor deva manter uma expansão controlada, haverá um aumento gradual tanto na produção nacional quanto nas exportações, impulsionado por uma demanda internacional consistente. Essa perspectiva reforça a necessidade de investimentos em tecnologia e gestão para aprimorar a eficiência da cadeia produtiva, permitindo que o Brasil continue a se destacar no mercado global de carne suína, mesmo com taxas de crescimento moderadas (2).

Em âmbito global, o consumo de carne suína tem apresentado tendências de crescimento significativas nos últimos anos. De acordo com a Organização para a Cooperação e Desenvolvimento Econômico (OCDE), o consumo per capita de carne suína na China foi de 23,68 kg em 2021, refletindo a importância dessa proteína na dieta chinesa (3). Projeções indicam que o consumo mundial de carne suína atingirá cerca de 131 milhões de toneladas em 2030, representando um aumento de 7,2% em relação a 2023 (4). No Brasil, o consumo per capita de carne suína também tem crescido de forma expressiva, com estimativas apontando um aumento de 1,8% em 2025 em relação a 2024 (5). Essas tendências refletem a crescente aceitação e demanda por carne suína tanto no mercado interno quanto no externo, impulsionando a produção e exportação brasileiras.

A cadeia produtiva da suinocultura no Brasil é composta por diversos elos interdependentes que garantem a eficiência e a qualidade da produção de carne suína. Essa estrutura abrange desde a produção de insumos básicos, como grãos utilizados na alimentação animal, até a chegada do produto final ao consumidor. Inicialmente, destacam-se os produtores de grãos e as fábricas de rações, responsáveis por fornecer alimentação adequada aos suínos. Em seguida, encontram-se os criadores, que podem atuar em sistemas de produção de ciclo completo ou especializados em determinadas fases, como a produção de leitões ou a terminação dos animais. Posteriormente, os transportadores asseguram o

deslocamento dos suínos até os abatedouros e frigoríficos, onde ocorre o processamento da carne. Além disso, a cadeia envolve segmentos de equipamentos, medicamentos, distribuição e, por fim, o consumidor final. A pesquisa e a inovação tecnológica também desempenham papel fundamental, apoiando produtores e agroindústrias na busca por soluções que garantam a qualidade da carne, conforme as normas de produção, sanidade e inspeção definidas pelo governo. (6)

O manejo adequado durante o embarque de suínos é uma etapa crítica no processo produtivo, pois influencia diretamente o bem-estar dos animais e a qualidade final da carne. O embarque representa o início do manejo pré-abate e expõe os suínos a diversos fatores estressantes, tais como mistura de animais, privação de alimento e água, condições ambientais desfavoráveis e densidade inadequada de carga. Esses fatores podem resultar em aumento significativo nos níveis de cortisol e ácido lático, prejudicando a qualidade da carne através de alterações fisiológicas como a carne PSE (pálida, mole e exsudativa). Assim, é essencial que todos os envolvidos no manejo sejam capacitados para reduzir o estresse animal, minimizando prejuízos econômicos e garantindo padrões éticos e de qualidade exigidos pelos consumidores. (7)

Nas últimas décadas, as tecnologias de visão computacional têm-se consolidado como ferramentas essenciais para a modernização da agropecuária, permitindo desde a identificação precoce de doenças em plantações até a contagem automatizada de frutos em pomares e a detecção de pragas. Por exemplo, estudos têm demonstrado que algoritmos baseados em redes neuronais convolucionais podem segmentar e identificar individualmente frutas em imagens, possibilitando estimativas precisas da produção e o planejamento estratégico do manejo agrícola (8). Essa mesma abordagem justifica-se para a contagem automática de suínos, pois facilita a identificação dos animais mesmo em condições adversas, como alta densidade e variações de iluminação. Ao adotar sistemas baseados em visão computacional, os produtores podem obter dados em tempo real com elevada precisão, promovendo melhorias significativas na eficiência operacional, redução dos erros na contagem e, consequentemente, o aprimoramento do bem-estar animal e da rentabilidade da produção.

## 1.1 Relevância e Justificativa

Diante dos desafios enfrentados no manejo tradicional, como erros na contagem manual, sobreposição de animais e variações nas condições ambientais, a automação do processo de contagem torna-se essencial para modernizar a suinocultura. A utilização de tecnologias de visão computacional, integrada a sistemas embarcados de baixo custo, permite reduzir erros operacionais, otimizar a gestão do rebanho e aprimorar a eficiência logística, contribuindo para a sustentabilidade econômica e o bem-estar animal. Além disso, a adoção de sistemas automatizados de identificação e rastreamento aumenta a

confiabilidade dos processos produtivos e proporciona controle rigoroso e transparente das operações, atendendo às exigências dos mercados internacionais por eficiência e precisão. Assim, esta pesquisa busca desenvolver soluções tecnológicas robustas que contribuam diretamente para a modernização da suinocultura brasileira, especialmente nos sistemas produtivos mais tecnificados e de larga escala, em um cenário de crescente demanda por produtos de qualidade e procedência ética.

Neste trabalho, são empregados métodos modernos de visão computacional baseados em aprendizado profundo, como as arquiteturas da família YOLO (You Only Look Once), utilizadas para detecção de objetos em tempo real, além de algoritmos de rastreamento múltiplo como SORT (Simple Online and Realtime Tracking) e ByteTrack. Esses métodos são implementados em sistemas embarcados de baixo custo, como o NVIDIA Jetson Nano, visando uma solução eficiente para contagem automatizada de suínos. A fundamentação técnica detalhada desses métodos é apresentada no Capítulo 3.

## 1.2 Objetivos do Trabalho

O objetivo geral deste trabalho é desenvolver e validar um sistema embarcado para identificação, rastreamento e contagem automática de suínos, utilizando o NVIDIA Jetson Nano como plataforma de processamento. Os objetivos específicos incluem: (i) implementar o algoritmo YOLOv12 para a detecção dos suínos, testando diferentes tamanhos de modelo para otimizar a relação entre precisão e desempenho; (ii) testar e comparar algoritmos de rastreamento – SORT e ByteTrack – para avaliar a manutenção da identidade dos animais em tempo real; (iii) realizar uma análise comparativa entre a contagem automática e a contagem manual de modo a identificar a solução mais eficaz para aplicações na suinocultura de médio e grande porte.

## 1.3 Estrutura da Dissertação

A presente dissertação está organizada em seis capítulos. O Capítulo 1 apresenta a introdução, contextualizando a suinocultura e justificando a necessidade do desenvolvimento do sistema proposto. No Capítulo 2, é realizada uma revisão bibliográfica que abrange os fundamentos teóricos dos algoritmos de detecção e rastreamento, além das tecnologias de visão computacional e sistemas embarcados. O Capítulo 3 descreve os materiais e métodos empregados na pesquisa, detalhando a configuração do NVIDIA Jetson Nano e a implementação dos modelos. O Capítulo 4 aborda o desenvolvimento e a implementação prática do sistema, enquanto o Capítulo 5 apresenta os resultados obtidos e a análise comparativa entre as abordagens testadas. Finalmente, o Capítulo 6 reúne as conclusões do estudo, discute as limitações encontradas e propõe direções para trabalhos futuros.

# 2 REVISÃO BIBLIOGRÁFICA

## 2.1 Introdução

O uso de tecnologias de visão computacional tem se tornado cada vez mais comum na pecuária de precisão, incluindo na monitoração e reconhecimento de comportamentos de animais. Neste capítulo, serão revisados artigos recentes que abordam o uso de técnicas de aprendizado de máquina e visão computacional para o reconhecimento de comportamentos de suínos em diferentes ambientes de criação.

Os estudos revisados utilizaram métodos como redes neuronais convolucionais (CNNs), que são modelos computacionais inspirados na estrutura do córtex visual e amplamente empregados na detecção de objetos em imagens. Também foram utilizados algoritmos de rastreamento, isto é, formulações que permitem acompanhar objetos ao longo de uma sequência de quadros, mantendo a identidade de cada indivíduo ao longo do tempo. Técnicas de segmentação foram aplicadas para separar regiões de interesse na imagem, enquanto a análise de comportamento e o reconhecimento de postura buscaram inferir ações e posições corporais dos suínos com base em seus movimentos observados.

## 2.2 Trabalhos correlatos

A contagem de suínos envolve diversos processos, seguindo etapas específicas: reconhecimento, rastreamento e contagem. Isso é devido ao fato de que para contar suínos é necessário reconhecer e rastrear o animal a fim de identificar os pontos no ambiente em que os mesmos estiveram. A ideia é não contar o animal mais de uma vez, por isso a necessidade de rotulá-los. Com isso, essa seção divide o estudo de trabalhos correlatos em tópicos baseados nas etapas da contagem.

### **2.2.1** Reconhecimento

Os artigos analisados nesta subseção utilizam algum tipo de melhoria no reconhecimento de suínos ou que não realizam algum tipo de rastreamento ou contagem dos animais.

Ahn et al. (9) propõem um método de detecção de suínos baseado em aprendizado profundo chamado *EnsemblePigDet*. O método consiste em dois modelos gerados e mesclados a partir de dois processamentos de imagem distintos, que servem como entrada para a geração dos modelos. O reconhecimento dos suínos é realizado utilizando YOLOv4 (10). Os resultados mostraram que o *EnsemblePigDet* é capaz de detectar suínos com precisão de até 95,36%, superando outros métodos de detecção baseados em aprendizado profundo.

Já Alameer et al. (11) propõem um método de detecção automatizada e quantificação do comportamento de contato entre suínos usando aprendizado profundo. O método utiliza uma rede neuronal baseada em YOLOv4 (10) modificada para detectar partes pequenas em imagens, adicionando uma décima camada convolucional na rede a fim de reconhecer cabeça e traseira de suínos, além de uma técnica de rastreamento baseada em Kalman (12) para rastreá-los. Os resultados mostraram que o método é capaz de detectar e rastrear suínos com precisão de até 96,2% e 95,3%, respectivamente.

No trabalho de Bhujel et al. (13) os autores propõem um sistema de monitoramento automatizado de atividades fisiológicas e comportamentais de suínos em diferentes concentrações de gases de efeito estufa. Eles usam uma rede neuronal convolucional treinada com imagens de suínos capturadas por câmeras de alta definição. O sistema foi testado em condições controladas de concentrações de gases de efeito estufa e mostrou-se capaz de detectar e monitorar atividades fisiológicas e comportamentais com precisão de até 96,3%.

Psota et al. (14) descreve um método baseado em aprendizado profundo para detectar e associar diferentes partes do corpo de suínos, como cabeça, tronco e cauda, em imagens e vídeos. Eles estabeleceram uma rede neuronal convolucional YOLO (15) baseada em pontos-chave para detectar partes distintas do corpo de suínos e destacaram a vantagem desta abordagem em relação às detecções de caixa de delimitação existentes. Eles relataram uma precisão de detecção de até 99% em ambientes controlados e de 91% em ambientes diversos, com iluminação distinta da que foi vista no treinamento.

O artigo de Tu et al. (16) descreve um sistema de detecção automatizada e segmentação de suínos em grupo. Eles usam uma rede neuronal chamada PigMS R-CNN, que é baseada na arquitetura MS R-CNN (*Mask Scoring* R-CNN) (17), para detectar e segmentar suínos em imagens. Essa rede é dividida em três etapas: uma rede residual de 101 camadas, combinada com a rede Piramidal de Características (*feature pyramid network*, FPN) (18) extrai as características de entrada; em seguida, essas características são usadas para gerar regiões de interesse e, para cada região, é possível obter informações sobre localização, classificação e segmentação dos suínos detectados. Para evitar que os suínos sejam perdidos ou que haja erros de detecção em áreas sobrepostas ou grudadas, é utilizado o método de supressão não-máxima suave (Soft-NMS) (19) ao invés de NMS (do inglês, *Non-maximum Suppression*) tradicional. Eles relatam um aumento de precisão, com um F1-score de 0,9374.

Wang et al. (20) descrevem uma rede neuronal chamada HRST, que é uma melhoria da arquitetura HRNet (do inglês, *High-Resolution Network*) (21), para detectar pontos de articulação em suínos. A rede HRST é uma rede HRNet com um bloco *Swin Transformer* (22). Os resultados mostraram que o CenterNet teve uma precisão média (AP) de 86,5% e o HRST teve uma precisão de 77,4% com uma velocidade de detecção de 40 imagens por segundo. Em comparação com o HRNet, o HRST melhorou a precisão em 6,8%,

enquanto reduziu o número de parâmetros e a quantidade calculada em 72,8% e 41,7%, respectivamente. O trabalho abre espaço para estudos relacionados ao cálculo do tamanho do corpo dos suínos via imagens.

O estudo de Ocepek et al. (23) descreve o desenvolvimento de um sistema de monitoramento automatizado para detecção de corpo, cabeça e cauda de suínos em fazendas intensivas. O projeto foi separado em duas etapas em que o primeiro foi o reconhecimento individual dos suínos em posições em pé e deitados, em grupos e suas partes do corpo, utilizando uma rede neuronal convolucional baseado na Mask R-CNN (24) e na rede Piramidal de Características (*feature pyramid network*, FPN) (18). A segunda etapa foi melhorar a detecção da postura da cauda (cauda reta e enrolada) durante a atividade (em pé ou movendo-se) utilizando YOLOv4 (10). O modelo reconheceu cada corpo de suíno individual com uma precisão de 96% relacionada ao limite de intersecção sobre união (IoU), enquanto a precisão para as caudas foi de 77% e para as cabeças foi de 66%, alcançando já precisão humana. A precisão da detecção de suínos em grupos foi a maior, enquanto a precisão de detecção de cabeça e cauda foi menor.

Em Song et al. (25), os autores propuseram um sistema de identificação de suínos baseado na rede neuronal convolucional ResNet (26). Foi proposto uma modificação na rede ResNet e aplicado para detectar individualmente suínos. O modelo desenvolvido capturou as características dos suínos aplicando conexões em camadas, e a capacidade de expressão de características foi melhorada adicionando um novo módulo residual. O número de camadas foi reduzido para minimizar a complexidade da rede. A precisão de treinamento e teste alcançou 98,2% e 96,4%, respectivamente, quando usando o modelo melhorado.

Já em Zhou (27) os autores propuseram uma melhoria no YOLOv5 (28), chamado de YOLOV5_Plus. Primeiramente, foi realizado um aumento de dados com translação de imagens, aumento de cor (aplicação de transformações de cor aleatórias nas imagens de treinamento), *mosaic* que é uma técnica de aumento de dados que consiste em dividir a imagem em pequenos pedaços reorganizando aleatoriamente para criar novas imagens e redimensionamento. Em seguida, é utilizado o YOLOV5_Plus para identificação de suínos, que consiste no YOLOv5 com um módulo de atenção chamado CBAM (do inglês, *Convolutional Block Attention Module*) (29). O YOLOV5_Plus pode alcançar uma precisão de 98,9%.

A subseção analisou diferentes artigos que propõem métodos de detecção e rastreamento de suínos com diferentes enfoques. Alguns dos métodos propostos usam aprendizado profundo, enquanto outros usam técnicas de rastreamento baseadas em Kalman. A precisão de identificação nos métodos propostos varia entre 95,3% e 99%, dependendo do método utilizado e do ambiente em que ele foi testado. Alguns dos métodos também têm capacidade de detectar e monitorar atividades fisiológicas e comportamentais de suínos.

**Tabela 1** – Comparação dos métodos de reconhecimento de suínos

| **Autor** | **Método** | **Precisão (%)** | **Observações** |
|---|---|---|---|
| Ahn et al. (9) | EnsemblePigDet + YOLOv4 | 95,36% | Dois modelos combinados para robustez |
| Alameer et al. (11) | YOLOv4 modificado + Kalman | 96,2% (detecção) / 95,3% (rastreamento) | Detecção de cabeça e traseira |
| Bhujel et al. (13) | CNN customizada | 96,3% | Detecção de atividades fisiológicas |
| Psota et al. (14) | YOLO baseado em pontos-chave | 99% (ambientes controlados) / 91% (diversos) | Detecção de partes corporais |
| Tu et al. (16) | PigMS R-CNN + Soft-NMS | F1-Score: 0,9374 | Detecção e segmentação de suínos em grupo |
| Wang et al. (20) | HRST (HRNet + Swin Transformer) | 77,4% (HRST) | Detecção de pontos de articulação |
| Ocepek et al. (23) | Mask R-CNN + YOLOv4 | 96% (corpo) / 77% (cauda) / 66% (cabeça) | Monitoramento de posturas |
| Song et al. (25) | ResNet modificada | 98,2% (treino) / 96,4% (teste) | Redução de complexidade da rede |
| Zhou (27) | YOLOv5_Plus + CBAM | 98,9% | Aumento de dados e atenção espacial |

### 2.2.2 Rastreamento

Nesta subseção, os artigos fazem o reconhecimento de suínos e o rastreamento do animal no ambiente, de tal forma que podemos analisar os métodos utilizados para avaliação.

Em Wutke et al. (30) foi proposto um método de detecção e rastreamento de suínos baseado em aprendizado profundo para quantificar o comportamento social de suínos. O método segue o modelo rastreamento-por-detecção (TBD, *tracking-by-detection* em inglês) utiliza uma rede neuronal convolucional com 25 camadas baseado em pontos-chave (proposto por Psota et al. (31)) para detectar partes distintas de corpos de suínos (orelha esquerda, orelha direita, ombro e ponto da cauda) e uma técnica de rastreamento filtrada utilizando o filtro de Kalman (12) para rastreá-los. Os resultados mostraram que o

método é capaz de detectar suínos com precisão de até 95,4% e rastreamento com MOTA (Multi-Object Tracking Accuracy) (32) de 94,4%, respectivamente.

No artigo de Cowton et al. (33) os autores descrevem um sistema de monitoramento automatizado para a localização, rastreamento e extração de métricas de comportamento de suínos individuais. Eles utilizam redes neuronais profundas, baseado no *Faster R-CNN* (34), para detectar suínos. O rastreamento foi avaliado utilizando dois métodos: rastreamento baseado na distância, que utiliza o SORT (*Simple Online and Real-time Tracking*) (35) combinado com filtro de Kalman (12) e algoritmo *Hungarian* (36), e o rastreamento baseado na distância e na análise visual que é baseado no algoritmo *DeepSORT* (37), utilizando também o filtro de Kalman e o algoritmo *Hungarian*. Eles relatam uma média da precisão média (*mean average precision*, mAP) de 0,901, MOTA (do inglês, *Multi-Object Tracking Accuracy*)(32) de 92% e IDF1 (*Identity F1-Score*) (38) de 73,4%.

No estudo de Xiao et al. (39) os autores propuseram um algoritmo para rastreamento de suínos chamado DT-ACR (do inglês, *Detection and Tracking based on a set of Association Rules with Constraint item*), que se baseia em regras bem definidas para determinar a posição da região de interesse. Quando o algoritmo falha em rastrear um objeto, ele ainda é capaz de continuar rastreando os objetos que não foram perdidos e recupera os objetos perdidos nas proximidades. Os experimentos mostraram que esse algoritmo é eficaz em rastrear objetos individuais em várias condições, incluindo cenas sem luz, cenas com brilho do sol, cenas de aderência e cenas de oclusão resultando em uma precisão geral de até 87,32% no rastreamento.

Em Gong et al. (40) os autores propõem uma melhoria do algoritmo IOU-Tracker (41). O IOU-Tracker é um algoritmo de rastreamento rápido baseado em um detector que calcula o valor de interseção do IOU (do inglês, Intersection over Union) da região de interesse antes e depois da imagem atual. A detecção dos suínos é realizada utilizando o sistema de aprendizagem profunda YOLOv5 (28). A melhoria no algoritmo fez com que, mesmo após uma falha na identificação da região de interesse, a identificação atribuída a ele não seja perdida. O método de rastreamento proposto resultou em um MOTA (do inglês, *Multi-Object Tracking Accuracy*)(32) de 97,60% e um MOTP (do inglês, *Multiple Object Tracking Precision*) (42) de 98,38% no rastreio de suínos.

A subseção apresenta vários estudos que propõem métodos de detecção e rastreamento de suínos baseados em aprendizado profundo. Esses métodos incluem técnicas como redes neuronais convolucionais, filtro de Kalman, rastreamento baseado em distância e análise visual, regras de associação, interseção sobre união, entre outros. Os resultados mostram que esses métodos são eficazes em detectar e rastrear suínos com precisões altas, mesmo em condições adversas como cenas sem luz, brilho do sol e oclusão.

**Tabela 2** – Comparação dos métodos de rastreamento de suínos

| **Autor** | **Método** | **MOTA (%)** | **Observações** |
|---|---|---|---|
| Wutke et al. (30) | CNN pontos-chave + Kalman | 94,4% | Detecção de orelha, ombro, cauda |
| Cowton et al. (33) | Faster R-CNN + SORT / DeepSORT | 92% (MOTA) / 73,4% (IDF1) | Rastreamento baseado em distância e visual |
| Xiao et al. (39) | DT-ACR (Regras de associação) | 87,32% | Recuperação de objetos perdidos |
| Gong et al. (40) | YOLOv5 + IOU-Tracker aprimorado | 97,6% (MOTA) / 98,38% (MOTP) | Manutenção de ID após falhas |

### **2.2.3** Contagem

Nesta subseção encontramos artigos que realizam o processo completo da contagem de suínos: reconhecimento, rastreamento e contagem.

Kim et al. (43) propõem um sistema de contagem de suínos baseado em visão computacional e aprendizado profundo que pode ser implementado em uma placa embarcada. O método proposto é dividido em 4 etapas: preprocessamento, detecção, rastreio e contagem. Eles utilizam uma rede neuronal convolucional chamada TinyYOLOv4 (10) para detectar suínos. A etapa de rastreio conta com o algoritmo LightSORT, proposto pelos autores e baseado no DeepSORT (37), focado em sistemas embarcados sem perda de acurácia. O sistema foi testado em vários cenários de criação de suínos e mostrou-se capaz de contar suínos com precisão de até 96,3%.

Em outro artigo (44) foi proposta uma técnica de contagem automatizada de suínos baseada em aprendizado profundo em que é utilizado uma versão modificada do *Counting Convolution neural Network* (*Counting* CNN) (45). A contagem é realizada através de um mapa de densidade, assim como é feito no *Counting* CNN. A contagem atingiu um erro absoluto médio (do inglês, *Mean Absolute Error*) de 1,67, sendo melhor que outros algoritmos ao realizar a comparação.

O *Counting* CNN (CCNN) proposto por Onoro-Rubio and López-Sastre (45) não utiliza uma técnica de rastreamento, sendo papel da rede neuronal convolucional de contar os objetos de interesse na imagem. O artigo apresenta dois modelos de aprendizado de máquina: o CCNN e o *Hydra* CNN. O CCNN é concebido como uma rede de regressão, na qual a rede aprende a associação entre a aparência de manchas de imagem e suas respectivas densidades de objetos. Já o *Hydra* CNN é um modelo de contagem sensível à escala, capaz de estimar densidades de objetos em cenários muito lotados em que não há informações geométricas disponíveis. Este modelo usa uma abordagem multiescala para

realizar a previsão de densidade final, aprendendo uma regressão não linear com base em manchas de imagem extraídas em diferentes escalas.

Em Hu et al. (46) os autores realizaram um pré-processamento utilizando o operador de Laplace a fim de reduzir os problemas de iluminação da imagem e desfoque da borda do alvo. A rede de extração de recursos usada na *Mask* R-CNN (24) foi melhorada utilizando a Resnet-152 (26) e a *Feature Pyramid Network* (FPN) (18). A rede foi aprimorada por meio de um caminho de melhoria de baixo para cima, que une diretamente os recursos de nível baixo com os de alto nível para melhorar a capacidade de reconhecimento de bordas desfocadas. Além disso, o processo de supressão não máxima e a função de perda foram otimizados para aprimorar a precisão da segmentação.

No artigo de Liu et al. (47) foi proposta uma rede semântica de segmentação e contagem para melhorar a precisão da segmentação e a eficiência da contagem de suínos na segmentação de imagens complexas chamada DeepLab V3+. Em primeiro lugar, um modelo de segmentação de imagem foi criado para identificar partes de uma imagem de suínos usando o DeepLab V3+. Isso foi feito para diminuir os custos de treinamento e obter características iniciais da imagem. Em seguida, um mecanismo de atenção simples foi adicionado para acelerar o cálculo das características da imagem e reduzir problemas relacionados a muitos parâmetros e redundância de características causados pela complexidade da rede. Por fim, um método cascata recursivo foi usado para melhorar a combinação de características de alta e baixa frequência para extrair informações semânticas significativas. A precisão de segmentação de imagens de suínos em ambiente simples chega a 99%.

Os trabalhos apresentados descrevem diferentes técnicas de contagem automatizada de suínos baseadas em aprendizado profundo e visão computacional. Cada um destes métodos utiliza uma abordagem diferente, incluindo o uso de redes neuronais convolucionais, operadores de Laplace, técnicas de segmentação de imagem e algoritmos de rastreio. Essas técnicas foram avaliadas e mostraram-se eficazes na contagem de suínos com uma elevada precisão, sendo capazes de contar suínos com erro absoluto médio de até 1,67 e precisão de até 96,3%. Estes métodos são promissores para aplicações práticas, especialmente na indústria de suinocultura.

**Tabela 3** – Comparação dos métodos de contagem de suínos

| Autor | Método | Precisão/Erro | Observações |
|---|---|---|---|
| Kim et al. (43) | TinyYOLOv4 + Light-SORT | 96,3% | Foco em sistemas embarcados |
| Tian et al. (44) | Counting CNN modificado | MAE = 1,67 | Contagem por mapa de densidade |
| Onoro-Rubio and López-Sastre (45) | Counting CNN / Hydra CNN | – | Modelos de regressão densidade |
| Hu et al. (46) | Mask R-CNN (ResNet-152 + FPN) | – | Aprimoramento de bordas e iluminação |
| Liu et al. (47) | DeepLab V3+ com atenção | 99% (segmentação) | Combinação de alta e baixa frequência |

### 2.2.4 Outros animais

Nesta subseção são abordados alguns trabalhos que realizam a identificação de outro tipos de animais, demonstrando as semelhanças e a possibilidade de aplicação do estudo neste trabalho com outras espécies, fomentando a evolução da pecuária de precisão.

O artigo de Qiao et al. (48) propõe um sistema de extração de contorno e segmentação de imagens em tempo real de bovinos utilizando *Mask* R-CNN (24). Neste artigo, é proposta uma solução para o problema de segmentação de instâncias de bovino e extração de contornos em um ambiente de confinamento real. A abordagem proposta envolve etapas como identificação dos quadros-chave do movimento dos animais, melhoria da imagem para diminuir o impacto da iluminação e sombras, segmentação do bovino e extração de seus contornos corporais. O método foi testado com sucesso em um conjunto de imagens complexas de bovinos, apresentando uma precisão de segmentação satisfatória com *Mean Pixel Accuracy* (MPA) de 0,92 e uma boa extração de contornos com *Average Distance Error* (ADE) de 33,56 *pixels*, melhor do que os métodos de segmentação de instâncias SharpMask (49) e DeepMask (50), que são outros métodos de segmentação aplicado diretamente nos *pixels* da imagem.

No trabalho de Jiang et al. (51), os autores propuseram um sistema de reconhecimento automático do comportamento de cabras confinadas em grupo utilizando aprendizado profundo através de imagens adquiridas em tempo real de câmeras localizadas no topo da área de confinamento. Neste estudo, quatro tipos de comportamento de cabras foram examinados. Isso foi feito analisando a relação entre a localização espacial dos delimitadores de identificação das cabras e as áreas de alimentação/bebida, bem como a quantidade de movimento da cabra identificada através dos centroides dos delimitadores de identificação quadro a quadro. O artigo compara a eficiência de reconhecimento

dos comportamentos entre os algoritmos de aprendizagem profunda *Faster R-CNN* (34), YOLOv3 (52) e YOLOv4 (10). Os experimentos revelaram que o YOLOv4 é melhor do que outros modelos no que diz respeito à velocidade e precisão na detecção de cabras. As acurácias médias de reconhecimento obtidas foram maior que 96% para as ações de comer, beber, comportamentos ativos e inativos, respectivamente, nos vídeos experimentais.

Em Li et al. (53) os autores desenvolveram um sistema de identificação e segmentação de imagens de frangos denominado MSAnet, com câmera no topo do ambiente confinado. Este estudo construiu um conjunto de dados de imagens de frangos para entender o comportamento dos frangos e aplicar análise inteligente. Depois, propôs-se uma abordagem eficaz para a segmentação de imagens de frangos, que pode ser aplicada a outras imagens de animais. Uma rede de codificador-decodificador baseada em multiescala foi utilizada para extrair recursos de vários níveis, e um módulo de atenção foi usado para aprimorar as características importantes. Além disso, uma função de perda combinada foi sugerida para melhorar a supervisão e resultar em uma segmentação mais eficaz. Os resultados experimentais mostraram a promissora performance da abordagem proposta na segmentação de imagens de frango ao ser comparada com os métodos de segmentação de imagens MSnet (54), MSCAnet (55) e MSEAnet (56).

O trabalho de Sun et al. (57) tem como objetivo usar o modelo *Faster R-CNN* (34) para detectar ovelhas recém-nascidas e cordeiros durante o parto. Para isso, o modelo foi treinado usando conjuntos de dados específicos de partos de ovelhas e baseado nas redes de extração de características ZFNet (58), VGG16Net (59) e no algoritmo Soft-NMS (19). A comparação dos resultados experimentais indica que o modelo *Faster R-CNN* combinado com o algoritmo Soft-NMS e a rede de extração de características VGG16Net é mais eficaz na detecção de partos de ovelhas.

A subseção apresenta estudos sobre a identificação de animais, como bovinos, cabras e frangos, em ambientes confinados utilizando técnicas de processamento de imagem e aprendizado profundo. Estes estudos propõem soluções para segmentação de imagem de animais e reconhecimento de seus comportamentos, buscando uma precisão satisfatória e melhorando o processo de análise inteligente na pecuária de precisão. Alguns dos métodos testados incluem *Mask* R-CNN (24), *Faster R-CNN* (34), YOLOv3 (52) e YOLOv4 (10), além de abordagens baseadas em codificador-decodificador multiescala com módulo de atenção e variações nas redes neuronais convolucionais para melhoria na identificação dos animais.

**Tabela 4** – Comparação de métodos aplicados a outros animais

| **Autor** | **Método** | **Precisão (%)** | **Observações** |
|---|---|---|---|
| Qiao et al. (48) | Mask R-CNN | 92% (MPA) | Segmentação de bovinos |
| Jiang et al. (51) | YOLOv4 (cabras) | >96% | Análise de comportamento em grupo |
| Li et al. (53) | MSAnet (frangos) | – | Segmentação multiescala com atenção |
| Sun et al. (57) | Faster R-CNN + Soft-NMS (ovelhas) | – | Detecção de partos de ovelhas |

## 2.3 Conclusão

A revisão da literatura demonstrou que as técnicas de detecção, rastreamento e contagem de suínos baseadas em aprendizado profundo representam o estado da arte na pecuária de precisão. Diversos estudos alcançaram altos índices de precisão, utilizando arquiteturas como YOLO, Faster R-CNN, Mask R-CNN, e variantes aprimoradas com mecanismos de atenção ou segmentação semântica. As abordagens também incluem rastreamento por filtros de Kalman, algoritmos como DeepSORT e IOU-Tracker, além de métodos de contagem baseados em mapas de densidade ou contagem direta por regressão.

Entretanto, apesar da eficácia desses métodos, poucos estudos integram todas as etapas (detecção, rastreamento e contagem) em sistemas completos e robustos, especialmente em cenários reais com restrições computacionais. A maioria das soluções propostas é voltada para ambientes controlados, com alta capacidade computacional, e não considera aspectos como repetibilidade de resultados, estabilidade em múltiplas execuções ou execução em tempo real em plataformas embarcadas de baixo custo.

Diante desse panorama, o presente trabalho busca preencher essa lacuna ao propor um sistema completo de contagem de suínos baseado em YOLOv12 e ByteTrack, com foco na execução embarcada em uma NVIDIA Jetson Nano. A proposta visa conciliar precisão, desempenho e estabilidade de contagem ao longo de execuções repetidas, promovendo avanços práticos e aplicáveis para a automação na suinocultura intensiva.

A Tabela 5 resume os principais trabalhos revisados, destacando suas abordagens e comparando diretamente com a solução proposta nesta dissertação.

**Tabela 5** – Resumo dos principais trabalhos relacionados

| **Trabalho** | **Detecção** | **Rastreamento** | **Contagem** | **Embarcado** | **Métricas** |
|---|---|---|---|---|---|
| Ahn et al. (9) | Sim | Não | Não | Não | Precisão: 95,36% |
| Alameer et al. (11) | Sim | Sim | Não | Não | Det.: 96,2%, Rast.: 95,3% |
| Cowton et al. (33) | Sim | Sim | Não | Não | MOTA: 92%, IDF1: 73,4% |
| Gong et al. (40) | Sim | Sim | Não | Não | MOTA: 97,6%, MOTP: 98,4% |
| Wutke et al. (30) | Sim | Sim | Não | Não | MOTA: 94,4% |
| Kim et al. (43) | Sim | Sim | Sim | Sim | Precisão: 96,3% |
| Tian et al. (44) | Sim | Não | Sim | Não | MAE: 1,67 |
| Liu et al. (47) | Sim | Não | Sim | Não | Segmentação: 99% |
| **Este trabalho** | Sim | Sim | Sim | Sim | Múltiplas execuções com métricas de ID, tempo e contagem |

# 3 TÉCNICAS AVANÇADAS EM DETECÇÃO E RASTREAMENTO DE OBJETOS

Este capítulo apresenta uma análise aprofundada dos métodos modernos aplicados à detecção e ao rastreamento de objetos em imagens e vídeos. Serão discutidos os fundamentos e os aprimoramentos das Redes Neuronais Convolucionais (CNN), o detector em tempo real YOLOv12, as métricas de avaliação de desempenho e os algoritmos de rastreamento ByteTrack e SORT. Cada seção detalha os conceitos teóricos, as fórmulas matemáticas e os passos dos algoritmos, oferecendo uma visão abrangente dos desafios e das soluções presentes nas aplicações de visão computacional.

## 3.1 Aspectos Técnicos e Fundamentação dos Métodos

A família de algoritmos *YOLO* (*You Only Look Once*) (15) revolucionou a detecção de objetos ao adotar uma abordagem única que integra a extração de características e a previsão de caixas delimitadoras em uma única etapa. Essa metodologia permite a detecção em tempo real com alta precisão, pois a rede divide a imagem em células e simultaneamente prevê a localização e a classe dos objetos, garantindo alta velocidade de processamento. Com a evolução desde o *YOLOv1* até as versões mais recentes, como o *YOLOv12* — que se baseia nas melhorias apresentadas por Tian et al. (60) — esse algoritmo mostra-se especialmente adequado para aplicações em ambientes dinâmicos e de alta densidade, como a contagem automática de suínos.

O NVIDIA Jetson Nano (61) foi escolhido como plataforma de computação embarcada devido à sua capacidade de acelerar operações de deep learning por meio de 128 núcleos da arquitetura Maxwell e um processador quad-core ARM. Embora seu desempenho não se equipare ao de GPUs de alto nível, o Jetson Nano apresenta uma solução de baixo custo e baixa potência (apenas 5W de consumo), ideal para aplicações em ambientes industriais e rurais, onde a portabilidade e a economia de recursos são essenciais para a implementação de sistemas autônomos de monitoramento.

No que diz respeito aos algoritmos de rastreamento, a continuidade e a confiabilidade na manutenção das identidades dos objetos ao longo de sequências de vídeo são fundamentais para uma contagem precisa. O algoritmo *SORT* (*Simple Online and Real-time Tracking*) (35), conhecido por sua simplicidade e baixo custo computacional, utiliza filtros de Kalman e o algoritmo húngaro para associação, mas pode enfrentar desafios em cenários com oclusões intensas. Adicionalmente, o *ByteTrack* (62) aproveita detecções de baixa confiança para melhorar o rastreamento em ambientes densos, garantindo uma identificação mais precisa mesmo em situações de sobreposição intensa .

## 3.2 Redes Neuronais Convolucionais (CNN)

### 3.2.1 Fundamentos e Arquitetura

As Redes Neuronais Convolucionais (CNN) são projetadas para processar dados com estrutura em grade, como imagens, aproveitando a correlação espacial dos pixels. Seu diferencial é a capacidade de extrair automaticamente características locais por meio de operações de convolução, o que reduz a necessidade de engenharia manual de *features* (63). Cada camada convolucional utiliza filtros (kernels) para capturar padrões como bordas, texturas e formas. Essa operação pode ser expressa pela equação:

$$S(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i+m, j+n) \cdot K(m,n), \tag{3.1}$$

na qual $I$ representa a imagem de entrada, $K$ é o kernel de dimensões $M \times N$ e $S$ é o mapa de características resultante.

Além da operação de convolução, outros blocos essenciais incluem:

- **Ativação:** Funções não-lineares (por exemplo, ReLU, definida por $f(x) = \max(0, x)$) que introduzem não-linearidades e permitem a modelagem de funções complexas (64).
- **Pooling:** Operações de *max pooling* ou *average pooling* que reduzem a dimensionalidade dos mapas de características, preservando as informações mais relevantes e contribuindo para a invariância a pequenas transformações.
- **Normalização:** Técnicas como Batch Normalization (65) aceleram o treinamento e estabilizam o processo de aprendizagem, normalizando as ativações de cada camada.

### 3.2.2 Treinamento e Backpropagation

O treinamento das CNNs se dá por meio do algoritmo de backpropagation, que utiliza a cadeia de regras da diferenciação para propagar os erros e ajustar os pesos dos filtros. Esse processo envolve a minimização de uma função de perda, que geralmente combina o erro de classificação com termos de regularização para evitar o overfitting. Uma das formas mais comuns de regularização é a **regularização L2**[1], que adiciona à função de perda um termo proporcional ao quadrado da magnitude dos pesos, penalizando valores muito altos e promovendo modelos mais simples e generalizáveis (63).

Além disso, técnicas como a utilização de otimizadores adaptativos, como o *Adam* (66), têm sido amplamente empregadas para acelerar a convergência e melhorar a estabilidade do treinamento.

[1] A regularização L2 é expressa como $\lambda \sum_i w_i^2$, onde $w_i$ são os pesos da rede e $\lambda$ é o coeficiente de regularização.

### 3.2.3 Arquiteturas de Referência

Diversas arquiteturas de CNN foram propostas e se tornaram referência em tarefas de visão computacional:

- **AlexNet:** Revolucionou a área ao demonstrar a eficácia das CNNs em larga escala, especialmente em competições de classificação de imagens (64).
- **VGGNet:** Destaca-se pela profundidade e simplicidade na organização das camadas, utilizando pequenos filtros de convolução (3x3) para extrair hierarquias de características.
- **ResNet:** Introduziu conexões residuais que permitem o treinamento de redes extremamente profundas, mitigando o problema de gradientes desaparecendo (26).

## 3.3 Detecção de Objetos com YOLO

### 3.3.1 Histórico e Evolução

O algoritmo *You Only Look Once* (YOLO), introduzido por Redmon et al. (67), reformulou a tarefa de detecção de objetos como um problema de regressão direta. Em vez de aplicar classificadores em regiões específicas da imagem (como no R-CNN), o YOLOv1 previu simultaneamente as caixas delimitadoras e suas respectivas classes a partir de toda a imagem de entrada. Essa abordagem permitiu grande ganho em velocidade, embora a precisão inicial fosse limitada, especialmente em objetos pequenos.

Na segunda versão, YOLOv2 ou YOLO9000 (15), melhorias significativas foram introduzidas, como o uso de *anchor boxes*, normalização em lotes (*batch normalization*) e o conceito de treinamento multi-rótulo. Além disso, essa versão combinou dados de detecção com dados de classificação, possibilitando a detecção de mais de 9000 categorias, uma inovação notável para a época.

Com o YOLOv3 (52), a precisão da detecção foi aprimorada por meio da adoção da arquitetura Darknet-53, uma rede residual profunda, e da introdução de previsões em múltiplas escalas. Essa versão conseguiu manter a velocidade do YOLO enquanto aumentava a robustez em objetos de tamanhos variados.

O YOLOv4 (10), desenvolvido por Bochkovskiy, incorporou várias otimizações modernas, como *Cross Stage Partial Connections* (CSP), *Weighted Residual Connections* (WRC), funções de ativação Mish, além de melhorias no treinamento que permitiram excelente equilíbrio entre desempenho e eficiência.

A partir do YOLOv5, desenvolvido pela Ultralytics, houve uma mudança no foco: apesar da ausência de publicação formal, o modelo se popularizou pela facilidade de uso, ampla documentação e suporte nativo a exportação para plataformas diversas (28).

As versões seguintes passaram a concentrar-se em aplicações específicas, com foco em desempenho real em ambientes produtivos.

O YOLOv6 (68) foi projetado para uso industrial, incorporando uma arquitetura eficiente e o uso de estratégias como *Bi-directional Concatenation* (BiC) e *Anchor-Aided Training* (AAT). Em seguida, o YOLOv7 (69) consolidou-se como referência em detectores de tempo real ao introduzir técnicas de regularização e treinamento chamadas *bag-of-freebies*, que melhoraram o desempenho sem aumentar o custo computacional.

O YOLOv8 trouxe mudanças estruturais importantes, como a adoção de cabeçalhos de detecção *anchor-free*, redesenhando partes da arquitetura com foco em generalização e velocidade. Embora sem artigo formal, sua documentação oficial fornece detalhes relevantes (70).

Mais recentemente, o YOLOv9 (71) introduziu o conceito de *Programmable Gradient Information* (PGI), permitindo que a rede aprenda de maneira programável e mais eficiente. Já o YOLOv10 (72) eliminou a etapa de *Non-Maximum Suppression* (NMS), tradicional em modelos anteriores, e otimizou a rede para tarefas ponta-a-ponta com menor custo computacional.

O YOLOv11 expandiu a arquitetura para múltiplas tarefas de visão computacional, como segmentação de instâncias e estimativa de pose, além de melhorias na eficiência computacional e na extração de características (73). Por fim, o YOLOv12 (60) introduziu mecanismos de atenção diretamente na arquitetura, permitindo maior precisão com tempo de inferência competitivo, combinando os benefícios das CNNs com atenção leve.

Ao longo de suas versões, o YOLO evoluiu de um modelo rápido porém limitado, para um ecossistema robusto e adaptável a diferentes tarefas, mantendo sempre o foco na eficiência de inferência. Essa evolução torna suas variantes especialmente atrativas para aplicações embarcadas e tempo real, como as implementadas nesta dissertação.

## 3.4 YOLOv12

O YOLOv12 representa um marco importante na evolução dos detectores de objetos em tempo real, integrando pela primeira vez, de forma eficaz, mecanismos de atenção dentro da arquitetura YOLO. Tradicionalmente, os detectores da série YOLO baseavam-se majoritariamente em redes convolucionais (CNN), devido à sua eficiência computacional superior quando comparada a mecanismos de atenção. No entanto, o YOLOv12 desafia este paradigma ao propor uma arquitetura centrada em atenção que mantém a alta velocidade dos modelos baseados em CNNs, ao mesmo tempo que colhe os benefícios de capacidade de modelagem dos transformers.

### 3.4.1 Contribuições Principais

As principais contribuições do YOLOv12 são:

- Proposição do **Area Attention Module (A2)**: Um módulo de atenção simples e eficiente que mantém um campo receptivo amplo enquanto reduz a complexidade computacional.
- Introdução do **R-ELAN (Residual Efficient Layer Aggregation Networks)**: Uma evolução da arquitetura ELAN, incorporando conexões residuais para superar desafios de otimização, especialmente em modelos de grande escala.
- **Aprimoramentos Arquiteturais** para compatibilidade com as exigências de velocidade do YOLO, incluindo:
  - Integração do *FlashAttention* para acelerar o processamento;
  - Remoção de *positional encoding* tradicional;
  - Ajuste da razão MLP de 4 para 1.2, melhor balanceando o custo computacional;
  - Aproveitamento máximo de operações convolucionais (`Conv2d` + BN).
- Criação de uma família de modelos escaláveis: YOLOv12 nano, small, medium, large e xlarge, atendendo a diferentes necessidades de precisão e velocidade.

### 3.4.2 Análise de Eficiência

Dois grandes desafios da adoção de atenção em detectores em tempo real são: a complexidade computacional quadrática do mecanismo de auto-atenção e os padrões ineficientes de acesso à memória.

O YOLOv12 resolve esses problemas através do uso do *Area Attention*, que segmenta o mapa de características de entrada em áreas simples (divisão uniforme), evitando partições complicadas como no *Swin Transformer*. Com isso, a complexidade computacional é drasticamente reduzida sem perda significativa de desempenho.

Além disso, o uso do *FlashAttention* otimiza o acesso à memória durante a computação da atenção, tornando o processo até 10 vezes mais eficiente.

### 3.4.3 Desempenho e Comparações

Nos benchmarks realizados no conjunto de dados MS COCO 2017, o YOLOv12 demonstrou desempenho superior em relação a todos os detectores em tempo real avaliados, como YOLOv10, YOLOv11 e RT-DETR.

Por exemplo, o modelo YOLOv12-N alcançou:

- **40.6% mAP** com latência de **1.64 ms** em GPU T4,
- superando o YOLOv10-N por 2.1% de mAP e o YOLOv11-N por 1.2%.

O modelo topo de linha, YOLOv12-X, atingiu:

- **55.2% mAP**,
- mantendo latência de **11.79 ms**,
- superando o YOLOv11-X por 0.6% mAP.

## 3.5 Métricas de Análise de Desempenho

A avaliação de modelos de detecção e rastreamento é fundamental para medir a eficácia e identificar áreas de melhoria. Entre as principais métricas utilizadas, destacam-se:

### 3.5.1 Intersection over Union (IoU)

A métrica IoU calcula a sobreposição entre a caixa predita e a caixa de referência (*ground truth*), sendo definida como:

$$\text{IoU} = \frac{\text{Área de Sobreposição}}{\text{Área de União}}. \tag{3.2}$$

Valores próximos de 1 indicam alta correspondência entre as caixas, sendo amplamente utilizada em desafios como o Pascal VOC (74).

### 3.5.2 Precisão, Revocação e F1-Score

Estas métricas avaliam a qualidade das detecções:

- **Precisão (Precision):** Proporção de detecções corretas em relação ao total de detecções realizadas:
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{3.3}$$
onde TP (True Positives) são os acertos e FP (False Positives) os falsos positivos.
- **Revocação (Recall):** Mede a capacidade do modelo em detectar todos os objetos reais:
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{3.4}$$
com FN (False Negatives) representando as detecções perdidas.
- **F1-Score:** A média harmônica entre precisão e revocação:
$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3.5}$$

### 3.5.3 Mean Average Precision (mAP)

O mAP agrega as precisões médias de todas as classes e é uma métrica chave para comparar a performance entre diferentes detectores. Essa métrica é calculada integrando a curva de precisão-revocação em múltiplos limiares de IoU (74).

## 3.6 Algoritmos de Rastreamento de Objetos

O rastreamento de múltiplos objetos (*Multi-Object Tracking* – MOT) visa manter a identidade dos objetos detectados ao longo de uma sequência de vídeo. Após a etapa de detecção em cada quadro, o desafio é determinar quais detecções pertencem ao mesmo objeto observado anteriormente. Esse processo é chamado de *associação de dados*, e resulta na criação de uma trajetória contínua para cada objeto.

Nesta seção, apresentam-se dois algoritmos de rastreamento amplamente utilizados em aplicações em tempo real: SORT e ByteTrack. Ambos utilizam modelos de movimento e algoritmos de associação para vincular detecções entre quadros consecutivos.

### 3.6.1 SORT (Simple Online and Realtime Tracking)

O algoritmo **SORT** foi proposto por Bewley et al. (35) com o objetivo de fornecer rastreamento eficiente e em tempo real, mantendo a simplicidade da implementação. Ele combina dois componentes principais:

- **Filtro de Kalman:** Modelo probabilístico usado para estimar o estado de cada objeto ao longo do tempo. O vetor de estado $\mathbf{x}$ geralmente representa a posição e a velocidade do objeto:
$$\mathbf{x} = [x, y, w, h, \dot{x}, \dot{y}]^T. \tag{3.6}$$
- **Algoritmo de Hungarian:** Utilizado para resolver o problema de associação entre as detecções do quadro atual e as previsões feitas pelo filtro de Kalman. Ele busca minimizar um custo (normalmente baseado na distância ou no *Intersection over Union* (IoU)) de modo ótimo e eficiente.

O funcionamento básico do SORT pode ser descrito em duas etapas:

1. **Predição:** Antes de receber as novas detecções, o estado de cada objeto é atualizado com base em seu modelo de movimento:
$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\,\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}\mathbf{u}_k, \tag{3.7}$$
onde $\mathbf{F}$ é a matriz de transição do modelo, $\mathbf{u}_k$ é o vetor de controle (geralmente zero) e $k$ indica o instante de tempo.

2. **Atualização:** Após a associação com as detecções, o filtro de Kalman é atualizado com a observação recebida:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}\right)^{-1}, \tag{3.8}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}), \tag{3.9}$$

onde $\mathbf{K}_k$ é o ganho de Kalman, $\mathbf{H}$ é a matriz de observação, $\mathbf{R}$ é a covariância do ruído, e $\mathbf{z}_k$ é a nova medição.

Apesar de sua eficácia e simplicidade, o SORT pode apresentar instabilidade em cenários com oclusões, detecções ausentes ou objetos muito próximos, resultando em trocas de identidade (*ID switches*).

### 3.6.2 ByteTrack

O **ByteTrack**, proposto por Zhou et al. (62), é uma evolução do SORT, projetada para lidar com cenários mais desafiadores e melhorar a robustez do rastreamento. Sua principal inovação está na utilização de detecções de baixa confiança, que normalmente são descartadas.

- **Motivação:** Em ambientes densos ou com oclusões frequentes, detecções com scores baixos muitas vezes correspondem a objetos válidos. Ignorá-las, como no SORT, pode resultar na perda de trajetórias.
- **Abordagem:** ByteTrack realiza a associação em duas fases: primeiro com detecções de alta confiança, depois tenta associar as detecções restantes (baixa confiança) a trilhas não associadas.

Seu funcionamento pode ser descrito como segue:

1. Prever a posição dos objetos ativos usando o filtro de Kalman.
2. Realizar a associação com as detecções de alta confiança utilizando o algoritmo de Hungarian.
3. Para as detecções de baixa confiança, repetir a associação com as trilhas restantes, de modo a recuperar trajetórias que poderiam ser descartadas.
4. Atualizar os estados e iniciar novas trilhas conforme necessário.

**Algorithm 1** Pseudo-código do ByteTrack

**Input:** Sequência de frames com detecções e scores
**Output:** Trajetórias atualizadas
Inicializar lista de trilhas
**for** *cada quadro* **do**
    Prever posição das trilhas com filtro de Kalman
    Associar detecções de alta confiança com as trilhas ativas
    Associar detecções de baixa confiança com trilhas não associadas
    Atualizar trilhas com base nas associações
    Criar novas trilhas para detecções restantes
**end**

O ByteTrack mostrou-se altamente eficaz em benchmarks como MOT17 (75) e MOT20 (76), superando modelos mais complexos sem utilizar descritores visuais. Isso o torna ideal para aplicações embarcadas e de baixo custo computacional.

### 3.6.3 Comparação entre SORT e ByteTrack

- **Simplicidade:** Ambos utilizam filtro de Kalman e algoritmo de Hungarian. SORT é mais simples, mas menos robusto.
- **Associação com baixa confiança:** ByteTrack diferencia-se por aproveitar detecções fracas, o que permite manter as identidades mesmo em oclusões temporárias.
- **Robustez:** Em ambientes com múltiplos objetos, ByteTrack apresenta menor taxa de *ID switches* e fragmentações de trilhas.
- **Desempenho computacional:** Ambos são rápidos e indicados para uso em tempo real, sendo adequados a plataformas como o Jetson Nano.

## 3.7 Descrição do Hardware Utilizado

### 3.7.1 NVIDIA Jetson Nano

O NVIDIA Jetson Nano (61) é uma plataforma de computação embarcada que oferece alta performance para aplicações de inteligência artificial. Neste trabalho, utilizamos o Jetson Nano devido à sua capacidade de processamento em tempo real e ao seu consumo energético reduzido. A seguir, listamos as principais especificações técnicas relevantes:

- Processador: Quad-core ARM Cortex-A57
- GPU: 128 núcleos Maxwell
- Memória: 4 GB LPDDR4

- Suporte a inferência de redes neuronais via TensorRT
- Consumo máximo de apenas 5W

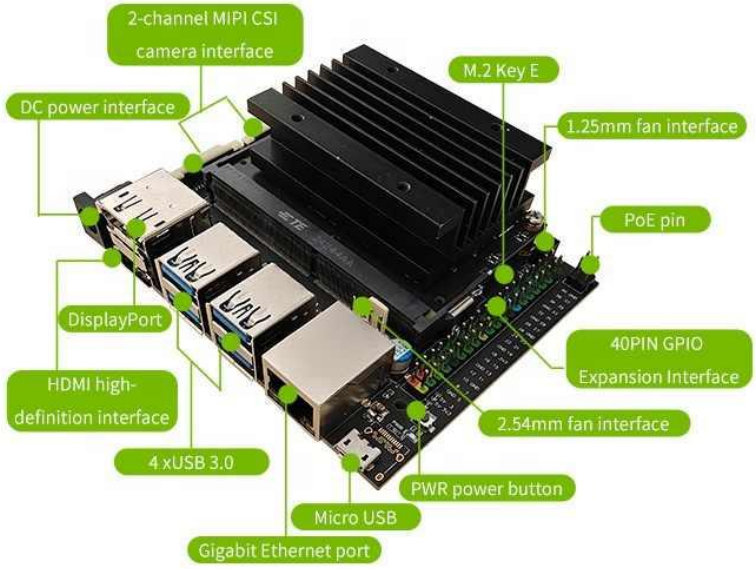


**Figura 1** – Nvidia Jetson Nano B01 4GB

Essas características permitem a execução eficiente dos modelos de detecção e dos algoritmos de rastreamento empregados nesta pesquisa.

## 3.8 Conclusão

Neste capítulo, foram abordados os fundamentos e os aprimoramentos de técnicas essenciais para a detecção e o rastreamento de objetos. As CNNs fornecem a base para a extração de características robustas (63, 64), enquanto os detectores da família YOLO, incluindo a proposta do YOLOv12 (15)(52)(60), demonstram a eficácia da detecção em tempo real. As métricas de avaliação, como IoU, precisão, revocação, F1-Score e mAP (74), permitem mensurar o desempenho de forma objetiva. Por fim, os algoritmos de rastreamento – ByteTrack (62) e SORT (35) – ilustram abordagens que variam desde métodos simples até técnicas avançadas que integram informações de movimento e aparência, essenciais para manter a consistência na identificação de objetos em vídeos. Essas técnicas são fundamentais para aplicações modernas em vigilância, veículos autônomos, robótica e diversas áreas emergentes da visão computacional.

# 4 METODOLOGIA

Neste capítulo, é apresentada a metodologia utilizada na pesquisa, que integra o uso de um NVIDIA Jetson Nano (61) para inferências com modelos YOLOv12 e a avaliação de dois algoritmos de rastreamento: SORT e ByteTrack. Adicionalmente, são comparados os modelos YOLOv12 (*nano*, *small*, *medium*, *large* e *xlarge*) quanto à acurácia na contagem automática versus a contagem manual, bem como o tempo de inferência.

## 4.1 Aquisição e Pré-processamento de Imagens

### 4.1.1 Coleta de Dados

A coleta de dados foi realizada em ambiente controlado, utilizando câmeras posicionadas estrategicamente para capturar imagens dos suínos no embarcador. Após instaladas as câmeras, foram capturados 6 vídeos de comprimentos diferentes que compõem também o *dataset* de validação da contagem automática. As imagens foram adquiridas variando o máximo possível o tipo de suíno, incluindo cor e tamanho do animal. A tabela 6 fornece as informações de cada vídeo

Na Figura 2 seguem alguns exemplos de imagens utilizadas no treinamento. Todas as imagens possuem 640×480 pixels.

**Tabela 6** – Informações dos vídeos

| **Vídeo** | **Contagem real** | **Duração (s)** |
|---|---|---|
| #1 | 15 | 59 |
| #2 | 15 | 61 |
| #3 | 15 | 62 |
| #4 | 14 | 79 |
| #5 | 16 | 67 |
| #6 | 15 | 62 |

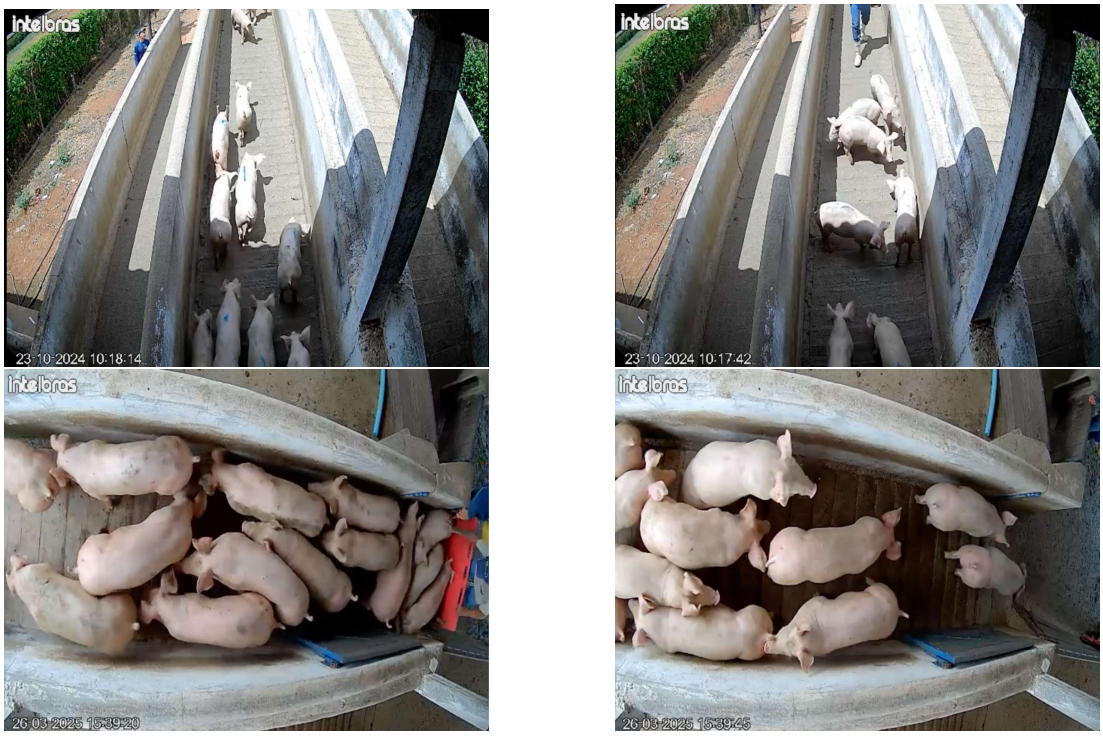


**Figura 2** – Exemplos de imagens do *dataset*

### 4.1.2 Organização do Dataset e Anotação

Esta etapa é crucial para adaptar o modelo às particularidades do ambiente e dos animais, buscando um equilíbrio entre acurácia e tempo de inferência. Inicialmente, foi realizada a anotação das imagens no dataset, selecionando as caixas delimitadoras nas regiões de interesse em cada imagem do *dataset*. Um exemplo desse processo pode ser visto na imagem 3.

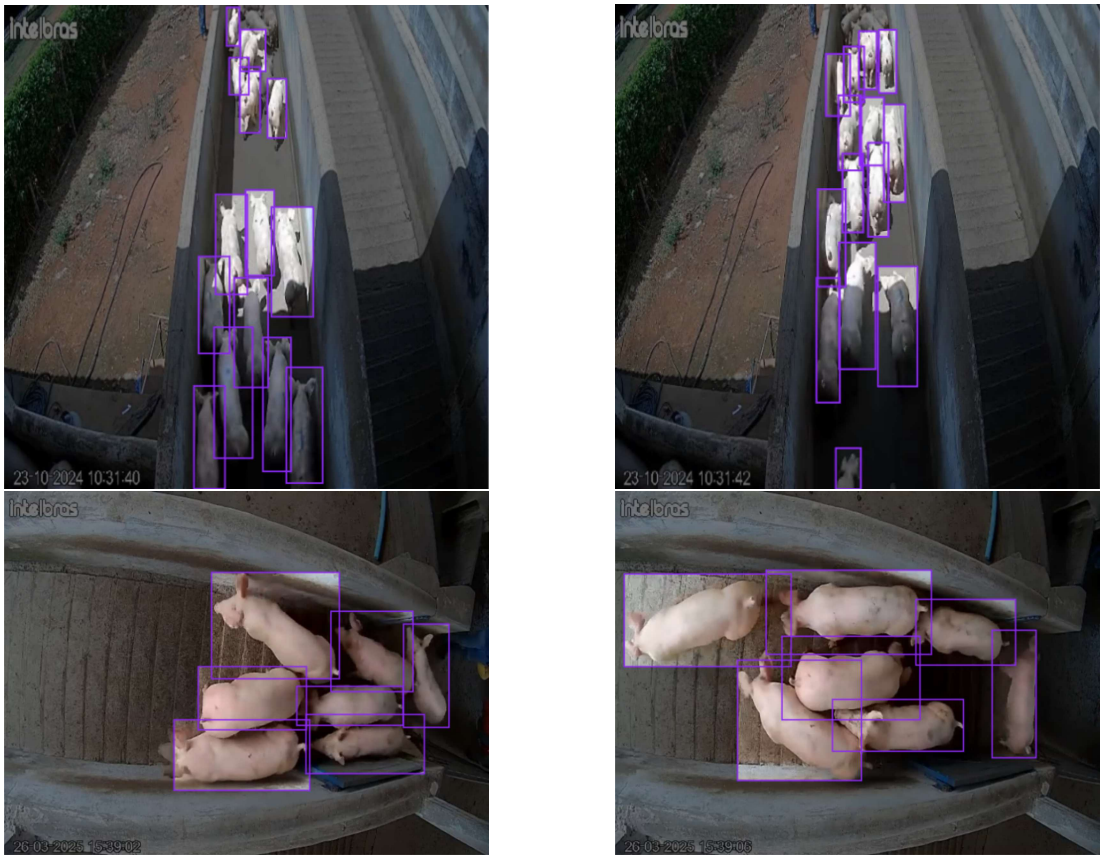


**Figura 3** – Anotação realizada no *dataset*

Ao todo foram realizadas anotações em 833 imagens extraídas dos vídeos adquiridos durante a realização de embarques em tempos distintos.

Em seguida, é realizado um processo chamado *Data Augmentation* que consiste em adicionar variações nas imagens como: rotações de 90°no sentido horário e anti-horário, rotação de 180°, pequenas rotações de 15°e adição de ruído. Com esse processo, o dataset passou a ter 1475 imagens. Além disso, o pré-processamento aplicado nas imagens consiste em apenas redimensioná-las para 640×640, a fim de normalizar os dados treinados com a etapa de inferência.

Para o treinamento, o *dataset* é dividido em 70% treinamento, 15% validação e 15% teste.

## 4.2 Modelos de Detecção de Objetos

### 4.2.1 Arquiteturas YOLOv12

Foram avaliadas diferentes versões do YOLOv12: **nano, small, medium, large e xlarge**. Cada modelo apresenta variações na quantidade de parâmetros e complexidade, impactando diretamente a acurácia e o tempo de inferência. Para cada modelo, foram definidos os parâmetros de treinamento, tais como taxa de aprendizado, número de épocas e tamanho dos lotes, considerando as especificidades do conjunto de dados utilizado.

### 4.2.2 Procedimento de Treinamento do YOLOv12

A configuração do treinamento deste modelo de detecção de objetos foi definida por um conjunto robusto de hiperparâmetros, cada um com um papel específico na aprendizagem e na generalização do modelo. A seguir, cada parâmetro técnico é descrito com referências que fundamentam suas escolhas na literatura.

#### Configurações Gerais

Parâmetros como o número de épocas (`epochs=100`), o tamanho do lote (`batch=16`) e a dimensão das imagens (`imgsz=640`) são fundamentais para garantir que o modelo tenha tempo e diversidade suficientes para aprender. A escolha do número de épocas e do tamanho do lote influencia diretamente a estabilidade e a velocidade da convergência, conforme discutido em (77).

#### Otimização

Em termos de otimização, embora os valores iniciais sugerissem uma taxa de aprendizado de 0.01 e um momentum de 0.937, o sistema optou por ajustar esses hiperparâmetros automaticamente e selecionar o otimizador AdamW, com uma taxa de aprendizado efetiva de 0.002 e momentum de 0.9. O AdamW é uma variação do Adam que desacopla o weight decay da atualização dos parâmetros, melhorando a regularização dos pesos (78). O weight decay (`weight_decay=0.0005`) atua penalizando grandes valores dos parâmetros, ajudando a prevenir o *overfitting*. Além disso, a estratégia de *warmup*, configurada para 3 épocas (`warmup_epochs=3.0`) com ajustes iniciais no momentum (`warmup_momentum=0.8`) e na taxa de aprendizado para os bias (`warmup_bias_lr=0.1`), permite uma estabilização inicial das atualizações, reduzindo oscilações indesejadas logo no início do treinamento (66).

#### Função de Perda

Os parâmetros que determinam os pesos relativos dos componentes da função de perda – `box (7.5)`, `cls (0.5)`, `dfl (1.5)`, `pose (12.0)` e `kobj (1.0)` – são configurados para equilibrar os diferentes aspectos do erro. O parâmetro `box` enfatiza a importância da regressão das caixas delimitadoras, enquanto `cls` define o peso da perda de classificação. A inclusão do Distribution Focal Loss (DFL) melhora a precisão na localização dos objetos, conforme explorado em (79), e `kobj` pondera a probabilidade de existência de um objeto na região proposta, auxiliando na distinção entre fundo e objeto.

#### Aumento de dados (*Data Augmentation*)

Diversas técnicas de *data augmentation* foram empregadas para aumentar a variabilidade dos dados e melhorar a robustez do modelo. Parâmetros como `hsv_h (0.015)`, `hsv_s`

`(0.7)` e `hsv_v (0.4)` ajustam os canais de matiz, saturação e brilho, respectivamente, alterando as características de cor das imagens. Transformações geométricas, controladas por `degrees (0.0)`, `translate (0.1)`, `scale (0.5)`, `shear (0.0)` e `perspective (0.0)`, permitem variações na orientação e no tamanho dos objetos. Técnicas como flip horizontal (`fliplr=0.5`) e *mosaic* (`mosaic=1.0`) ampliam ainda mais a diversidade dos dados. Embora o método *mixup* esteja desativado (`mixup=0.0`), ele tem sido reconhecido por sua eficácia na regularização dos modelos (80). Adicionalmente, o uso de *auto_augment* baseado em RandAugment (81) e o *random erasing* (`erasing=0.4`, conforme (82)) contribuem para simular diferentes condições de oclusão e variações de iluminação.

Outros Aspectos Técnicos

O uso do *Automatic Mixed Precision* (AMP, `amp=True`) acelera o treinamento e reduz o consumo de memória ao combinar cálculos em precisão simples e dupla, conforme evidenciado em (83). Parâmetros como `dropout (0.0)` podem ser ajustados para prevenir *overfitting*.

## 4.3 Algoritmos de Rastreamento de Objetos

## 4.4 SORT - *Simple Online and Realtime Tracking*

### 4.4.1 Parâmetros e Aspectos Técnicos do Rastreador SORT Implementado

A implementação do algoritmo SORT (*Simple Online and Realtime Tracking*) (35) adotada neste trabalho foi cuidadosamente ajustada para atender ao cenário específico de rastreamento de suínos em ambientes confinados. A seguir, descrevem-se os principais aspectos técnicos e parâmetros configurados no desenvolvimento.

### 4.4.2 Modelo de Estado Kalman Filter

Cada objeto rastreado é modelado por um filtro de Kalman com sete variáveis de estado:

- $x$, $y$ — centro da caixa delimitadora;
- $a$ — razão de aspecto da caixa;
- $h$ — altura da caixa;
- $v_x$, $v_y$, $v_h$ — velocidades correspondentes às variáveis de posição e altura.

O filtro foi inicializado com as seguintes configurações:

- Matriz de transição de estado adaptada para incluir componentes de velocidade.

- Matriz de ruído do processo com valor $1 \times 10^{-2}$.
- Matriz de ruído de medição definida como $1 \times 10^{-1}$.
- Matriz de covariância do erro inicializada com identidade, para garantir rápida convergência nas primeiras iterações.

Os valores foram ajustados empiricamente com base em experimentos preliminares realizados no ambiente de teste, equilibrando a suavização das trajetórias com a responsividade frente a movimentos rápidos.

### 4.4.3 Critério de Associação Detecção-Rastro

A associação entre as detecções fornecidas pela rede YOLOv12 e os rastros ativos foi realizada com uma estratégia híbrida baseada na métrica de *Intersection over Union* (IoU) e distância euclidiana entre os centros das caixas delimitadoras.

O critério de associação considera:

- Associações com **IoU** $> 0.3$ são aceitas diretamente.
- Para detecções com IoU intermediária ($0.1 <$ IoU $\leq 0.3$), é aplicada uma restrição adicional de distância máxima de **50 pixels** entre os centros das caixas. A limitação de 50 pixels foi definida considerando a largura média dos animais nas imagens capturadas, o que permite discriminar entre objetos adjacentes sem penalizar pequenas imprecisões nas detecções.
- A pontuação de associação é calculada como: score $= \frac{\text{IoU}}{1+\text{distância}}$, favorecendo pares com alta sobreposição e proximidade espacial.

Este critério composto foi fundamental para lidar com as sobreposições comuns em cenários de corredor estreito, mantendo robustez contra ruídos ocasionais da detecção.

### 4.4.4 Gerenciamento de Rastros

Do ponto de vista da longevidade dos rastros:

- O parâmetro **max_age** foi fixado em **3 frames**, limitando o tempo que um rastro pode permanecer sem atualização antes de ser descartado. Considerando a taxa de 30 quadros por segundo, um valor de 3 quadros (0,1 segundos) para $\mathbf{max}_a ge foi suficiente para manter a continuidade dedos$

Essa abordagem evita a criação e remoção excessiva de rastros causados por ruídos momentâneos ou detecções instáveis.

### 4.4.5 Inicialização de Novos Rastros

As detecções não associadas a rastros existentes são imediatamente promovidas a novos rastros ativos, utilizando a mesma estrutura de *Kalman Filter*, garantindo responsividade do sistema frente a novos alvos.

### 4.4.6 Resumo das Configurações

**Tabela 7** – Parâmetros da implementação do SORT

| **Parâmetro** | **Valor** |
|---|---|
| Dimensão do estado | 7 variáveis (posição, tamanho, velocidade) |
| Ruído do processo | $1 \times 10^{-2}$ |
| Ruído da medição | $1 \times 10^{-1}$ |
| IoU mínimo para associação | 0.3 (associação direta) |
| IoU mínimo com restrição de distância | 0.1 (distância $< 50$ pixels) |
| Max_age (idade máxima sem atualização) | 3 frames |
| Idade mínima para validação | 2 frames |
| Pontuação de associação | $\frac{\text{IoU}}{1+\text{distância}}$ |

## 4.5 Parâmetros e Aspectos Técnicos do Rastreador ByteTrack Implementado

O rastreador ByteTrack, na implementação desenvolvida neste trabalho, foi configurado para maximizar a estabilidade das identificações e minimizar a fragmentação de rastros em cenários desafiadores, como corredores estreitos e movimentação intensa de suínos. A seguir, descrevem-se os principais parâmetros e considerações técnicas da implementação.

### 4.5.1 Configuração dos Limiares de Confiança

A arquitetura do ByteTrack separa as detecções em duas categorias: detecções de alta e de baixa confiança, o que permite manter rastros mesmo diante de flutuações na confiabilidade dos detectores. Os limiares utilizados foram:

- **track_thresh = 0.5** — Limiar inferior para aceitar detecções como candidatas a associação com rastros ativos.
- **high_thresh = 0.6** — Limiar superior para inicialização de novos rastros.
- **match_thresh = 0.8** — Limiar utilizado durante o processo de associação de detecções com rastros existentes por meio da distância de IoU.

Os limiares foram testados em diferentes combinações, e os valores adotados resultaram no menor número de fragmentações e troca de IDs nas sequências avaliadas.

### 4.5.2 Estratégia de Associação Hierárquica

O ByteTrack adota uma associação hierárquica em múltiplas etapas:

1. **Associação inicial com detecções de alta confiança**: utilizando a distância de IoU e otimização pelo algoritmo de associação linear (LAPJV (84)).
2. **Reassociação com detecções de baixa confiança**: reutilizando rastros não associados na primeira etapa, aumentando a robustez.
3. **Confirmação de rastros não confirmados**: processamento específico para rastros recém-criados, geralmente ainda instáveis.

A associação em duas etapas se mostrou essencial, dado que a detecção de suínos em movimento apresentou variações de confiança devido à oclusão parcial entre os indivíduos.

### 4.5.3 Gestão de Rastros e Buffer Temporal

A gestão do tempo de vida dos rastros foi configurada da seguinte maneira:

- **max_time_lost = (frame_rate / 30) × track_buffer** — Número máximo de frames durante os quais um rastro pode permanecer não atualizado antes de ser definitivamente removido.

Esta flexibilidade temporal permite lidar melhor com oclusões momentâneas e entradas ou saídas parciais do campo de visão da câmera.

### 4.5.4 Algoritmo de Associação

Para a correspondência entre detecções e rastros, é empregada a métrica de distância baseada no **IoU invertido** $(1 - \text{IoU})$. A matriz de custos assim calculada é otimizada utilizando o algoritmo *LAPJV* (Jonker-Volgenant), garantindo atribuição eficiente e precisa, mesmo em casos de múltiplos candidatos concorrentes.

A distância de IoU utilizada adota um limiar dinâmico por etapa:

- **0.8** para associações principais.
- **0.5** para segunda associação com detecções de baixa confiança.
- **0.7** para associação de rastros não confirmados.

### 4.5.5 Remoção de Duplicações de Rastros

Para evitar múltiplos rastros redundantes sobre o mesmo animal, a implementação incorpora uma etapa final de remoção de duplicatas. Rastros duplicados são detectados por sobreposição excessiva (IoU abaixo de 0.15) e são eliminados com base no tempo de vida dos rastros, mantendo o mais longevo.

### 4.5.6 Resumo das Configurações

**Tabela 8** – Parâmetros da implementação do ByteTrack

| **Parâmetro** | **Valor** |
|---|---|
| Limiar de rastreamento inicial (`track_thresh`) | 0.5 |
| Limiar alto para nova associação (`high_thresh`) | 0.6 |
| Limiar de associação principal (`match_thresh`) | 0.8 |
| Tempo máximo perdido (`max_time_lost`) | (frame_rate / 30) × track_buffer |
| Algoritmo de associação | LAPJV (84) |
| Critério de associação | $1 - \text{IoU}$ |
| Limiar duplicação de rastros | $\text{IoU} < 0.15$ |
| Associação hierárquica | Sim (alta e baixa confiança) |

### 4.5.7 Critérios de Seleção

A escolha destes algoritmos baseou-se em sua representatividade na literatura e na capacidade de operar em tempo real, característica essencial para aplicações práticas no monitoramento de suínos.

Além disso, tendo em vista que os suínos são muito idênticos entre si, foram descartados algoritmos baseados em aparência, como: DeepSORT, FairMOT e BoT-SORT. Esses algoritmos utilizam algum modelo de extração de características, porém, devido à complexidade no objeto de interesse, o tempo de execução da *pipeline* é aumentado desnecessariamente, sem as vantagens do uso do extrator de características.

**Algorithm 2** Algoritmo de Contagem de Suínos

**Input:** Lista de rastros detectados no quadro atual
**Output:** Contagem acumulada de suínos
Inicializar variáveis globais: $count \leftarrow 0$, $trackHistory \leftarrow \emptyset$, $recentlyLostTracks \leftarrow \emptyset$

**Função** `update(`$tracks$`):`
    Atualizar rastros perdidos: incrementar $frames\_since\_lost$, remover se $> 10$ quadros
    **foreach** $track \in tracks$ **do**
        Calcular centro $(current\_x, current\_y)$ do rastro
        Verificar proximidade com outros centros: ignorar se $distância < min\_dist$
        **if** $track \notin trackHistory$ **then**
            Tentar recuperar histórico de rastros perdidos
            Verificar entrada pela margem da cena: atualizar $count$ se necessário
            Criar novo histórico para o track
        **else**
            Determinar lado atual e anterior do track em relação à linha
            Se mudou de lado e não foi contado: atualizar $count$, marcar como contado
            Atualizar posição anterior no histórico
        **end**
    **end**
    Atualizar histórico: mover rastros ausentes para $recentlyLostTracks$

## 4.6 Metodologia de Contagem de Suínos

O algoritmo de contagem implementado na classe `PigCounter` foi desenvolvido para realizar a contagem automática de suínos em corredores estreitos, considerando direções de movimento, aproximação espacial e a robustez contra múltiplas detecções e rastreamentos temporariamente perdidos.

O método se baseia na definição de uma linha imaginária vertical no espaço da imagem (*counting line*), posicionada em uma coordenada `line_x`. A passagem dos animais por esta linha, de acordo com a direção do movimento, incrementa ou decrementa a contagem total acumulada.

### 4.6.1 Controle e Filtros Espaciais

Para cada quadro processado, é calculado o centro de massa de cada objeto rastreado utilizando as coordenadas do retângulo delimitador (bounding box). Objetos com centros excessivamente próximos entre si são descartados da contagem momentânea, utilizando uma distância mínima de exclusão de `30 pixels`, evitando múltiplas contagens de animais muito próximos ou de detecções redundantes.

Além disso, é considerado uma *margem de cena* (`scene_margin`) de `150 pixels`, tanto na entrada quanto na saída do campo de visão, prevenindo a contagem de objetos que desapareçam ou surjam muito próximos das extremidades da imagem, o que é comum

em ambientes restritos.

### 4.6.2 Histórico de Rastreamento

O algoritmo mantém um histórico dos rastros ativos e dos recentemente perdidos. Rastros perdidos são mantidos em memória por até 10 quadros, permitindo reconexões curtas e aumentando a robustez do sistema frente a falhas temporárias de detecção.

Quando um objeto reaparece após ter sido perdido, o histórico é consultado para inferir sua direção anterior e a possível necessidade de ajustar a contagem, levando em consideração as coordenadas passadas e atuais do objeto.

### 4.6.3 Detecção de Transições

Para cada rastro, o algoritmo determina de que lado da linha de contagem o centro do objeto se encontra. A mudança de lado indica uma possível transição válida:

- Caso o objeto tenha se deslocado da direita para a esquerda (de `last_x > line_x` para `current_x < line_x`), a contagem é incrementada.
- Caso contrário, se o movimento foi da esquerda para a direita (de `last_x < line_x` para `current_x > line_x`), a contagem é decrementada.

A fim de evitar múltiplas contagens para o mesmo rastro, cada objeto possui um estado booleano, que é ativado após uma transição ser detectada, sendo reinicializado apenas quando o movimento do objeto estabiliza de um lado da linha.

### 4.6.4 Gestão de Rastros Perdidos

Quando um rastro ativo não está presente na nova lista de objetos detectados, ele é transferido para a estrutura de rastros recentemente perdidos. Caso este rastro não retorne após 10 quadros, ele é permanentemente descartado da memória, prevenindo consumo excessivo de recursos.

### 4.6.5 Visualização da Linha de Contagem

Para fins de depuração e análise visual, a linha de contagem e suas áreas adjacentes são desenhadas no quadro de saída:

- A linha central de contagem é desenhada em vermelho.
- As áreas de proximidade são preenchidas com transparência em verde, indicando as zonas de entrada e saída da cena, com base no parâmetro `scene_margin`.

Esta visualização auxilia na validação e no entendimento do funcionamento do algoritmo durante a execução. A Figura 4 apresenta uma imagem da cena de contagem, com a linha de contagem e a zona de observação para detecções perdidas.

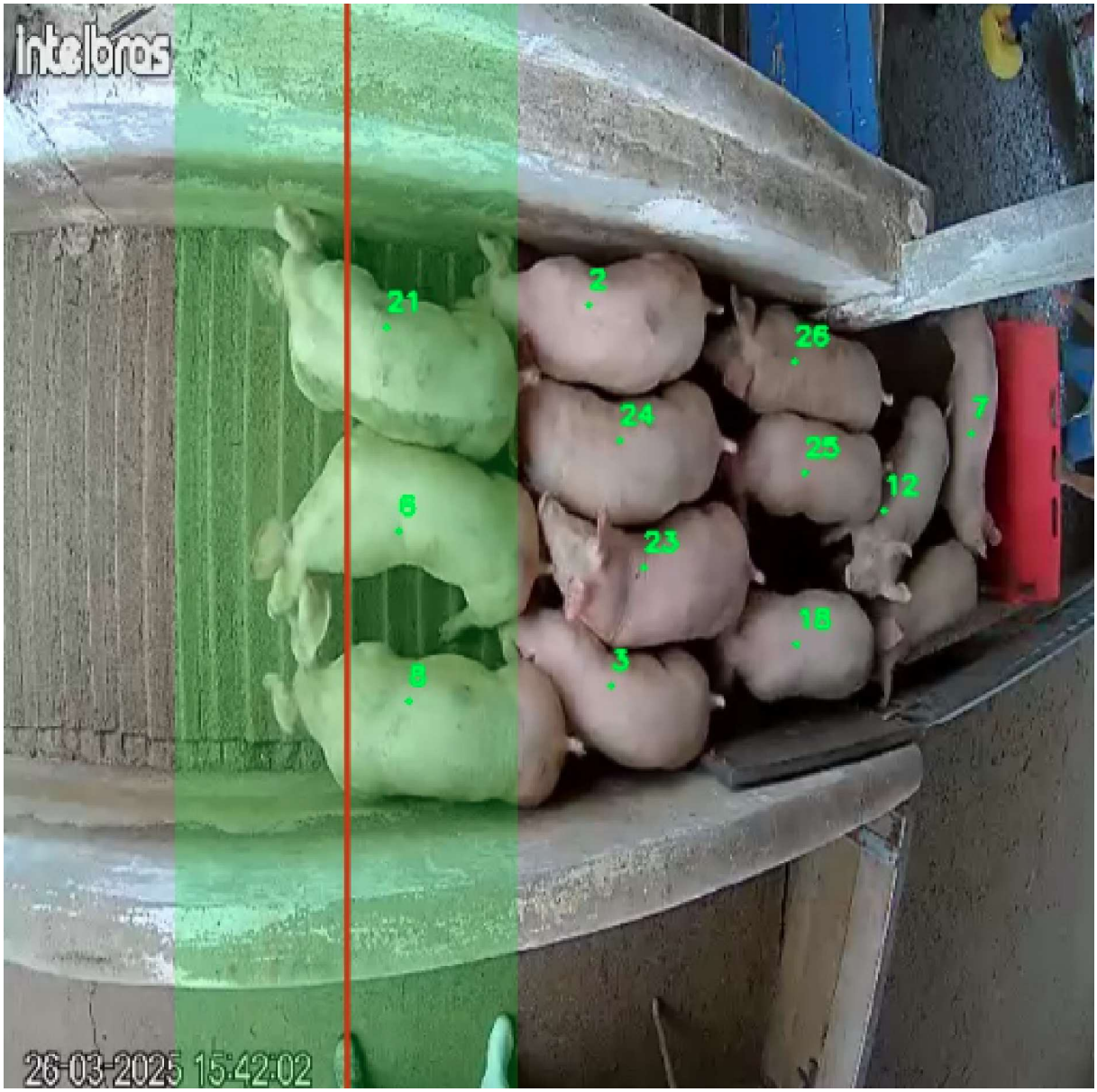


**Figura 4** – Imagem da cena de contagem.

### 4.6.6 Resumo do Fluxo de Processamento

O fluxo geral do algoritmo pode ser resumido nas seguintes etapas:

1. Atualização dos rastros recentemente perdidos.
2. Filtragem dos rastros ativos com base na distância mínima entre centros.
3. Verificação e atualização do histórico de cada rastro.
4. Detecção de transições pela linha de contagem e ajuste da contagem acumulada.

5. Atualização do estado de rastros ativos e gerenciamento dos rastros perdidos.

6. Visualização opcional da linha de contagem e das áreas de interesse.

A implementação deste método permite uma contagem eficiente e robusta de suínos, mesmo em cenários desafiadores com sobreposição de objetos e perdas temporárias de rastreamento.

## 4.7 Procedimentos de Implementação

### 4.7.1 Integração dos Modelos de Detecção e Rastreamento

A implementação consiste na integração dos modelos YOLOv12 com cada um dos algoritmos de rastreamento. Essa integração é realizada por meio de um *pipeline* de processamento, onde a etapa de detecção gera as caixas delimitadoras e a etapa de rastreamento realiza a associação das detecções entre os *frames*.

### 4.7.2 Pipeline de Processamento

O fluxo de processamento adotado é o seguinte:

**Algorithm 3** Thread 1: Pipeline de Processamento

**Input:** Buffer de frame (1 frame)
**Output:** Contagem de suínos, métricas de desempenho

```
while vídeo não finalizado do
    if buffer não vazio then
        Obter o frame atual do buffer
        Pré-processar o frame
        Executar inferência com modelo YOLOv12
        Aplicar o algoritmo de rastreamento
        Atualizar a contagem de suínos
        Registrar tempos de processamento e métricas
        Limpar o buffer após processamento
    else
        Aguarda novo frame no buffer
    end
end
Armazenar métricas finais da execução
```

**Algorithm 4** Thread 2: Captura e Envio de Frames

**Input:** Vídeo de entrada
**Output:** Frames inseridos no buffer
**while** *vídeo não finalizado* **do**
    Capturar o próximo frame do vídeo
    **if** *buffer vazio* **then**
        Inserir o frame capturado no buffer
    **else**
        Descartar o frame capturado para manter o processamento em tempo real
    **end**
**end**
Finalizar captura de vídeo

A aquisição de vídeo para a *pipeline* de processamento simula o ambiente real, de modo que durante a inferência os *frames* atuais são descartados.

Em virtude da natureza não determinística do experimento, a qual abrange tanto o processo de inferência quanto a aquisição das imagens, a *pipeline* de processamento é executada em 10 repetições para cada configuração avaliada dos modelos de detecção e rastreamento. Posteriormente, uma vez identificada a combinação otimizada para ambos os módulos, a *pipeline* é submetida a 600 execuções, com o objetivo de permitir a extração de métricas estatísticas robustas acerca do desempenho global do sistema.

### 4.7.3 Ambiente de Desenvolvimento e Frameworks Utilizados

Visando garantir a execução eficiente da *pipeline* em tempo real, o desenvolvimento foi inteiramente realizado na linguagem C++, aproveitando as bibliotecas NVIDIA TensorRT e CUDA para aceleração do processamento, em uma plataforma embarcada Jetson Nano.

A escolha do C++ como linguagem base justifica-se pela necessidade de controle refinado sobre os recursos computacionais e pela otimização de baixo nível que a linguagem proporciona. Tais características são essenciais para aplicações embarcadas, onde desempenho e eficiência energética são fatores críticos. Além disso, o C++ oferece excelente integração com as bibliotecas da NVIDIA, permitindo extrair o máximo desempenho da GPU integrada ao Jetson Nano.

#### *4.7.3.1* TensorRT

O TensorRT é uma biblioteca da NVIDIA especializada na otimização e execução de redes neuronais profundas em hardware acelerado por GPU. Entre as otimizações realizadas, destacam-se:

- **Fusão de camadas**: Redução da latência por meio da combinação de múltiplas operações em uma única execução.
- **Precisão mista (FP16 e INT8)**: Utilização de tipos de dados com menor precisão para acelerar o processamento, mantendo níveis adequados de acurácia.
- **Eliminação de operações redundantes**: Simplificação do grafo computacional da rede neuronal.
- **Kernel tuning**: Seleção automática dos kernels CUDA mais adequados para cada operação de inferência.

No contexto deste trabalho, o TensorRT foi empregado para converter e executar o modelo YOLOv12 de forma otimizada, garantindo alta taxa de processamento e baixa latência na plataforma Jetson Nano.

#### *4.7.3.2* CUDA

O CUDA (Compute Unified Device Architecture) é a plataforma de computação paralela da NVIDIA, que permite utilizar a GPU para acelerar tarefas de propósito geral. No projeto desenvolvido, o CUDA foi aplicada em múltiplas etapas do processamento, incluindo:

- Pré-processamento das imagens, como redimensionamento e normalização, realizado diretamente na GPU para minimizar a sobrecarga da CPU.
- Execução acelerada das inferências via integração com TensorRT.
- Processamento paralelo de detecções e dados de rastreamento, garantindo o desempenho necessário para manter o fluxo em tempo real.

A utilização do CUDA permitiu explorar plenamente a arquitetura de 128 núcleos CUDA da GPU integrada ao Jetson Nano, resultando em significativas reduções nos tempos de inferência e processamento global.

## 4.8 Métricas de Avaliação

### **4.8.1** Contagem Manual versus Automática

Para validar a eficácia do sistema, foi realizada uma contagem manual dos suínos em amostras selecionadas, que serviu de parâmetro para comparar com a contagem obtida automaticamente pelo sistema.

### 4.8.2 Tempo de Inferência

O tempo de inferência foi medido para cada combinação de modelo YOLOv12 e algoritmo de rastreamento. Essa métrica é fundamental para avaliar a viabilidade do sistema em aplicações em tempo real.

### 4.8.3 Número de IDs gerados

O número de IDs gerados pelo algoritmo de rastreamento. Quanto mais próximo o número de IDs gerados com o número da contagem de animais, mais preciso é o algoritmo de rastreamento e resiliente a oclusões. É importante também que seja maior do que a contagem real.

## 4.9 Procedimentos Experimentais

### 4.9.1 Configuração dos Testes

Os experimentos foram conduzidos em ambiente controlado, com variações de cenários para simular diferentes condições reais de monitoramento. Cada configuração experimental incluiu:

- Número definido de testes para garantir a reprodutibilidade dos resultados;
- Variação dos modelos YOLOv12 e algoritmos de rastreamento, permitindo uma análise comparativa abrangente;
- Registro detalhado dos resultados, incluindo contagens, tempos de inferência e eventuais inconsistências.

## 4.10 Considerações Éticas e de Bem-Estar Animal

A pesquisa respeita os preceitos éticos e garante o bem-estar dos animais envolvidos. Foram adotadas as seguintes medidas:

- Minimização do estresse e desconforto durante a coleta de dados;
- Adoção de práticas que assegurem o manejo adequado dos suínos;
- Conformidade com as diretrizes estabelecidas pelos comitês de ética em pesquisa animal.

## 4.11 Resumo da Metodologia

Neste capítulo, detalhamos a abordagem metodológica adotada, abrangendo desde a descrição do hardware e dos procedimentos de aquisição e pré-processamento de imagens até a implementação dos modelos de detecção e algoritmos de rastreamento. Foram estabelecidos critérios de avaliação rigorosos, incluindo a comparação entre contagem manual e automática, bem como a medição do tempo de inferência. A inclusão das etapas de organização do *dataset*, treinamento do YOLOv12, juntamente com a análise de 6 vídeos de tamanhos diferentes, fornece uma base sólida para a análise crítica dos resultados obtidos e fundamenta a discussão apresentada nos capítulos seguintes.

# 5 RESULTADOS E DISCUSSÕES

Neste capítulo, são apresentados e analisados os resultados obtidos com o sistema de identificação, rastreamento e contagem de suínos em tempo real. Os dados aqui expostos são simulados, servindo como base para a validação e comparação dos métodos empregados.

## 5.1 Treinamento dos Modelos YOLOv12 (*nano*, *small*, *medium*, *large* e *xlarge*)

Durante o treinamento dos modelos de detecção de objetos, diversos indicadores são coletados para avaliar o aprendizado e a capacidade de generalização do modelo. O indicador *losses* de treinamento, o *box_loss*, *cls_loss* e o *dfl_loss* medem o quão bem o modelo está ajustando as caixas delimitadoras, classificando corretamente os objetos e refinando a distribuição das bordas, respectivamente. Já as métricas de validação (*val/box_loss*, *val/cls_loss*, *val/dfl_loss*) seguem a mesma lógica, porém são aplicadas a um conjunto de dados que não participa do treinamento, permitindo observar se há sinais de *overfitting* ou subajuste. Em termos de desempenho, analisam-se também a *precision* e o *recall*, que indicam, respectivamente, a taxa de detecções corretas em relação a todas as predições positivas e a proporção de detecções corretas entre todos os objetos existentes. Por fim, o *mAP@50* (média de *precision* para um limiar de 50% de sobreposição) e o *mAP@50-95* (média em múltiplos limiares, de 50% a 95%) fornecem uma visão global de quão bem as caixas preditas coincidem com as reais, sintetizando a qualidade final do modelo. As Figuras 5, 6, 7, 8 e 9 apresentam as métricas do treinamento do YOLOv12 para os diferentes modelos.

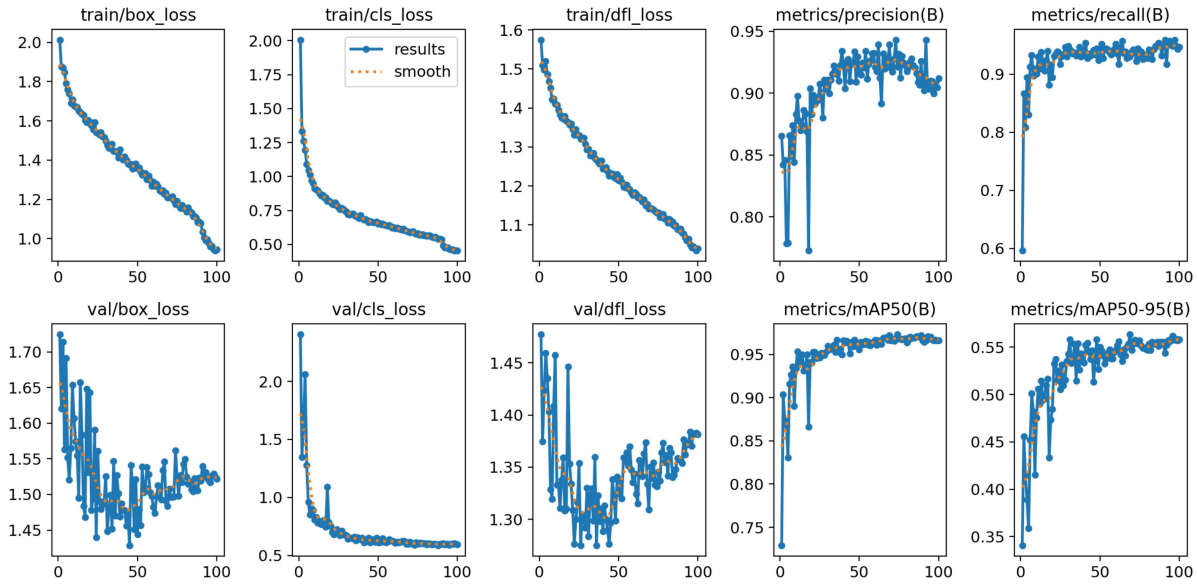


**Figura 5** – Métricas de treinamento do modelo *nano*.

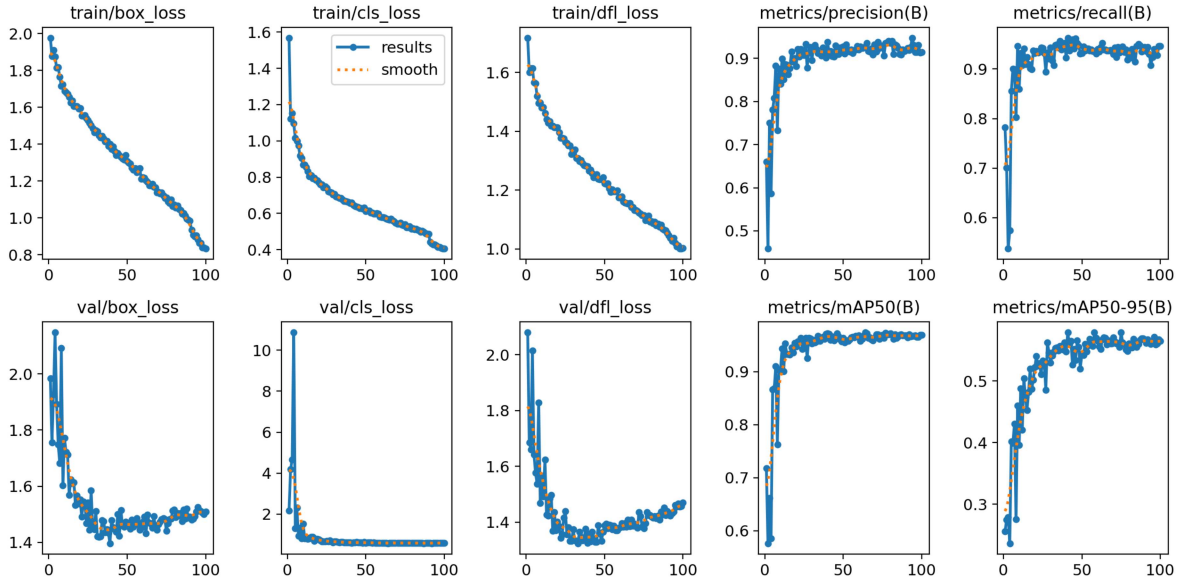


**Figura 6** – Métricas de treinamento do modelo *small*.

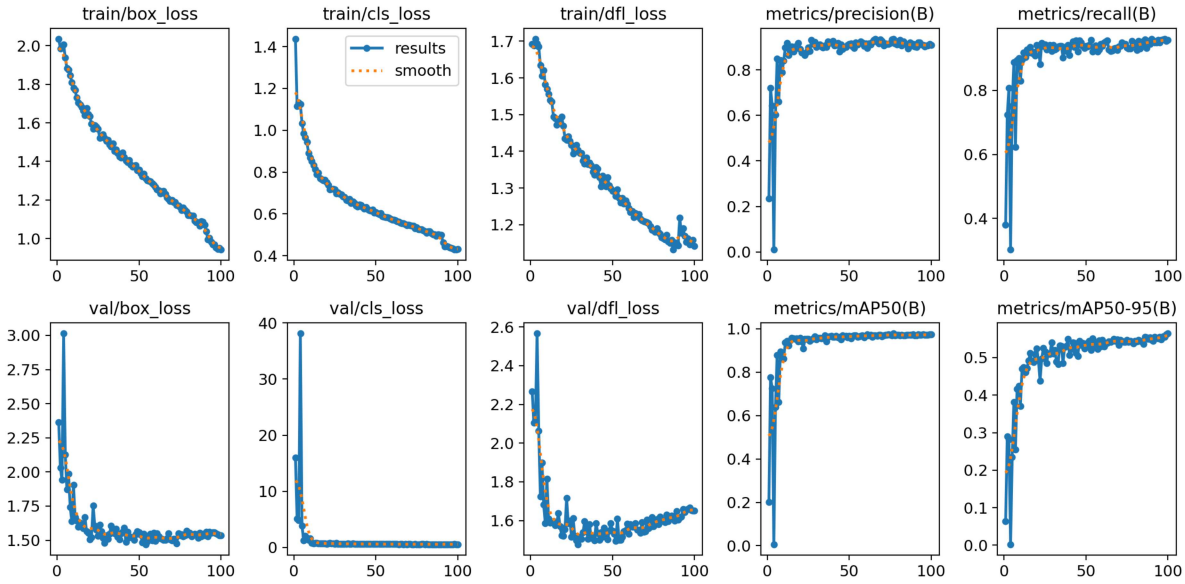


**Figura 7** – Métricas de treinamento do modelo *medium*.

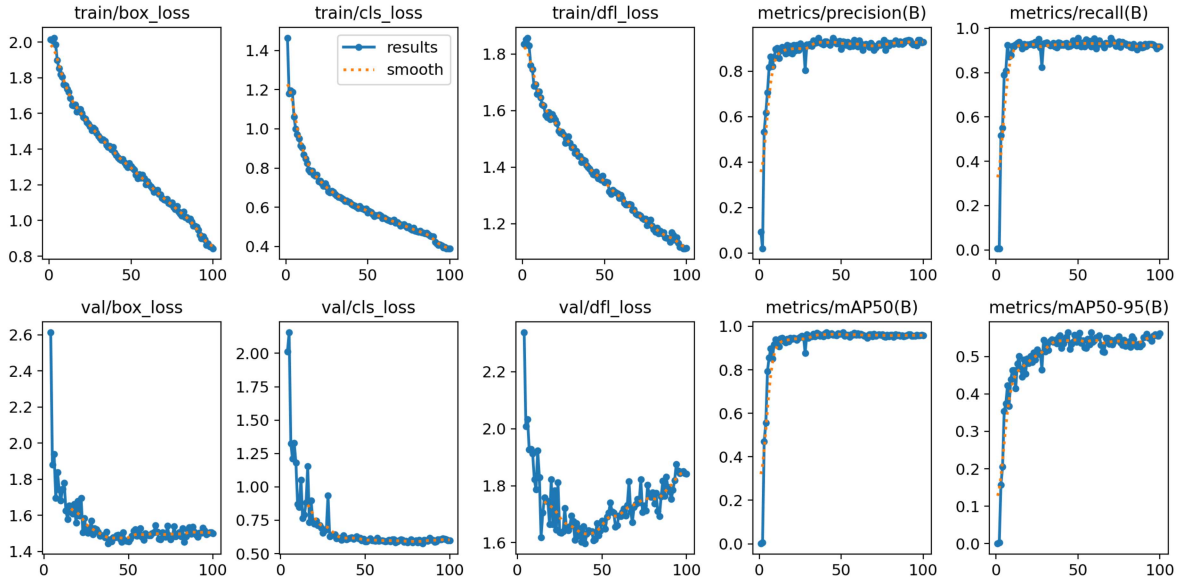


**Figura 8** – Métricas de treinamento do modelo *large*.

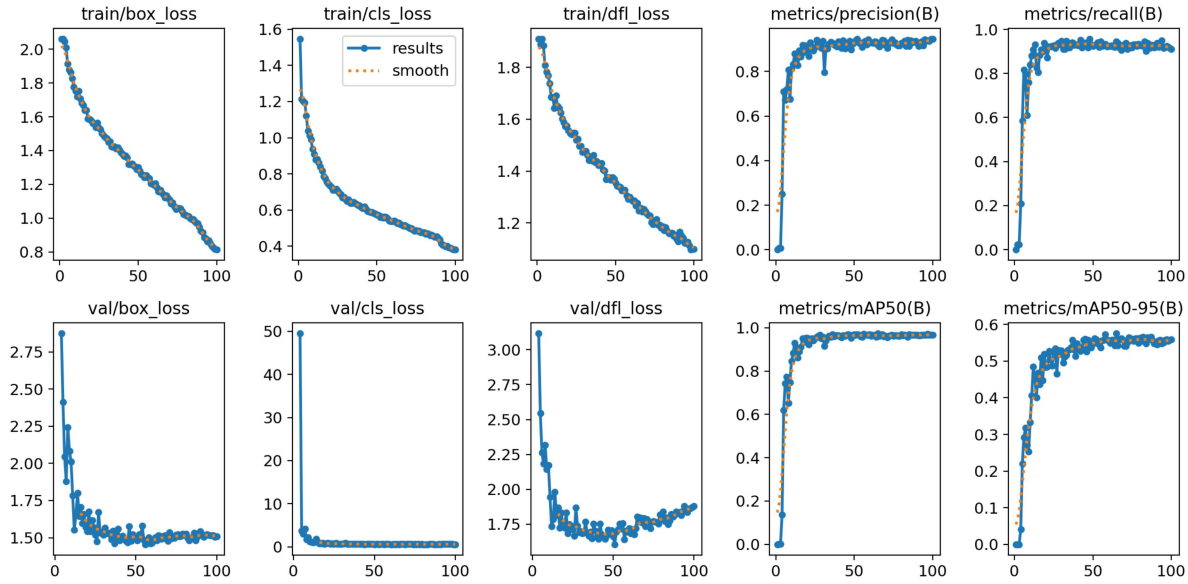


**Figura 9** – Métricas de treinamento do modelo *xlarge*.

Os resultados de treinamento dos modelos YOLOv12 (*nano*, *small*, *medium*, *large* e *xlarge*) mostram que todos apresentam curvas de perdas (tanto de treinamento quanto de validação) em constante declínio, indicando uma convergência adequada. De modo geral, observa-se que as versões menores (*nano* e *small*) tendem a convergir de forma mais rápida, mas com valores de perda finais ligeiramente superiores, sugerindo menor capacidade de generalização. Por outro lado, os modelos maiores (*large* e *xlarge*) mantêm uma queda mais estável nas curvas de validação e costumam atingir níveis mais baixos de perda, embora isso possa exigir mais tempo de treinamento e maior poder de processamento. O modelo

*medium*, por sua vez, oferece um equilíbrio interessante, alcançando bom desempenho sem demandar recursos tão elevados.

No que diz respeito às métricas de desempenho (*precision*, *recall*, mAP@50 e mAP@50-95), percebe-se que as arquiteturas mais robustas (*large* e *xlarge*) obtêm valores finais mais altos, refletindo uma detecção e localização mais precisas. Ainda assim, as versões *nano* e *small* exibem resultados competitivos, sendo vantajosas em cenários com restrições de hardware ou necessidade de inferência em tempo real, graças ao menor número de parâmetros e à maior velocidade de processamento. Nesse sentido, a escolha do modelo ideal deve considerar não apenas as métricas de acurácia, mas também fatores como capacidade computacional disponível, demanda de FPS (quadros por segundo) e requisitos de latência da aplicação.

## 5.2 Análise Geral dos Modelos e *Trackers*

### 5.2.1 Análise por Vídeo

A seguir, são apresentados os resultados individualizados por vídeo. Cada figura contém quatro gráficos representando as métricas médias de contagem na ida, na volta, a soma total e o MaxTrackID.

#### *5.2.1.1* Vídeo 1

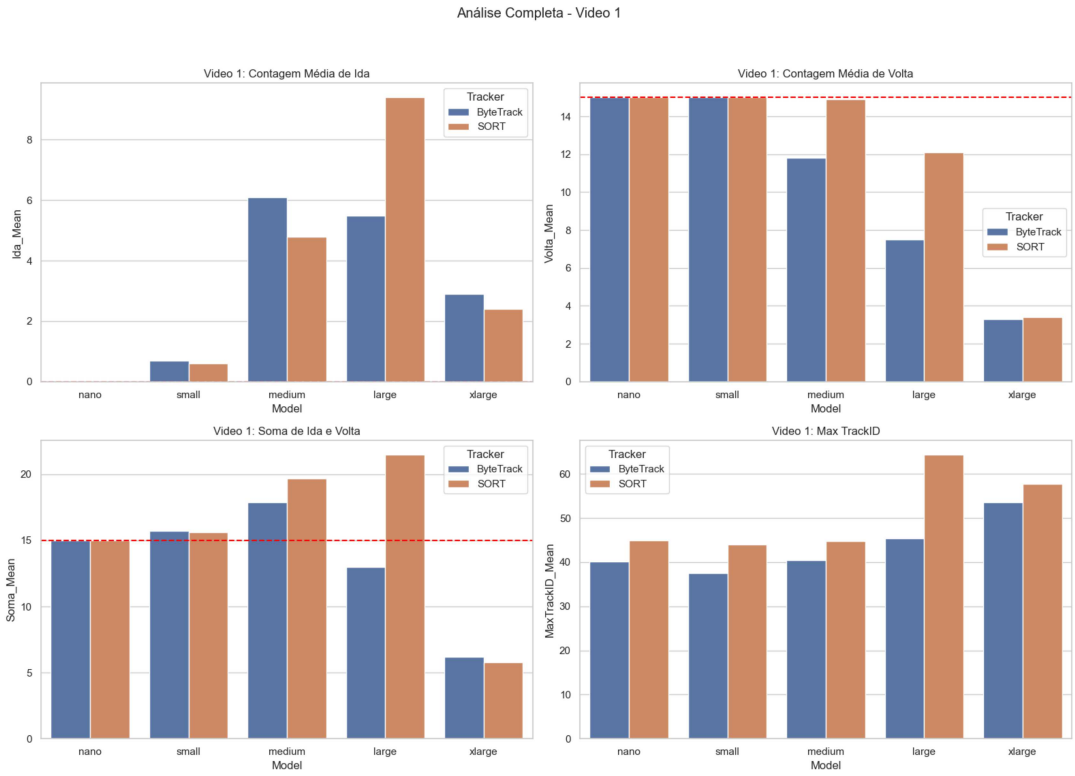


**Figura 10** – Análise completa - Vídeo 1.

No vídeo 1, que contém 15 animais, a soma média das contagens de ida e volta para o modelo *nano* com ByteTrack aproximou-se do total correto, com valores de ida praticamente nulos e a volta mantendo-se no patamar ideal de 15. Este desempenho reforça a robustez desta combinação de modelo e *tracker* para cenários simples e com fluxo controlado.

Modelos maiores, em especial *large* e *xlarge*, apresentaram aumento na contagem de ida e MaxTrackID, refletindo comportamento menos desejável. O SORT, por sua vez, manteve-se inferior, com desvios mais pronunciados.

#### *5.2.1.2* Vídeo 2

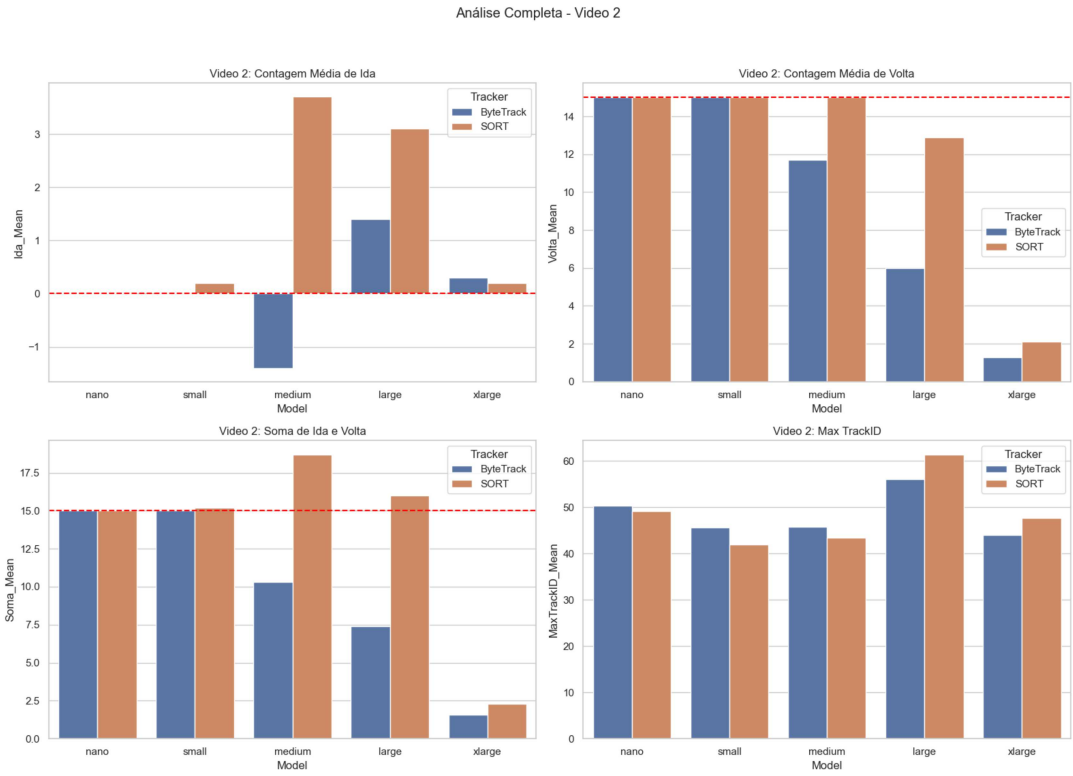


**Figura 11** – Análise completa - Vídeo 2.

Para o vídeo 2, com 15 animais, a tendência observada anteriormente se repetiu. O ByteTrack com modelo *nano* apresentou excelente performance, mantendo a soma próxima ao esperado e contagem de ida praticamente nula. Por outro lado, modelos superiores indicaram elevação nas contagens de ida e MaxTrackID.

O *tracker* SORT novamente apresentou desempenho inferior, evidenciado por maiores desvios nas métricas analisadas.

#### *5.2.1.3* Vídeo 3

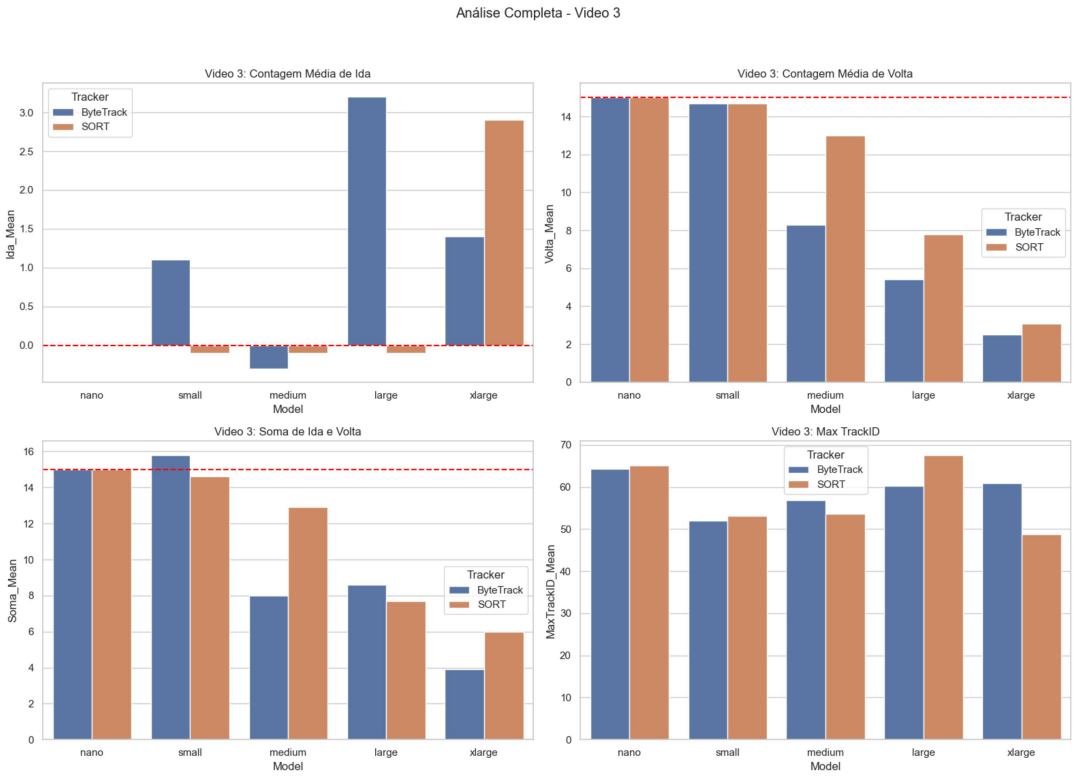


**Figura 12** – Análise completa - Vídeo 3.

Os resultados do vídeo 3, também com 15 animais, reforçam a superioridade do ByteTrack com modelo *nano*. A contagem de ida foi mínima e a soma de ida e volta permaneceu próxima ao ideal. Este vídeo apresentou um dos menores valores de MaxTrackID para essa configuração, indicando excelente capacidade de rastreamento e baixa taxa de geração de IDs desnecessários.

Modelos mais robustos e o *tracker*SORT mantiveram a tendência de pior desempenho observada nos vídeos anteriores.

#### *5.2.1.4* Vídeo 4

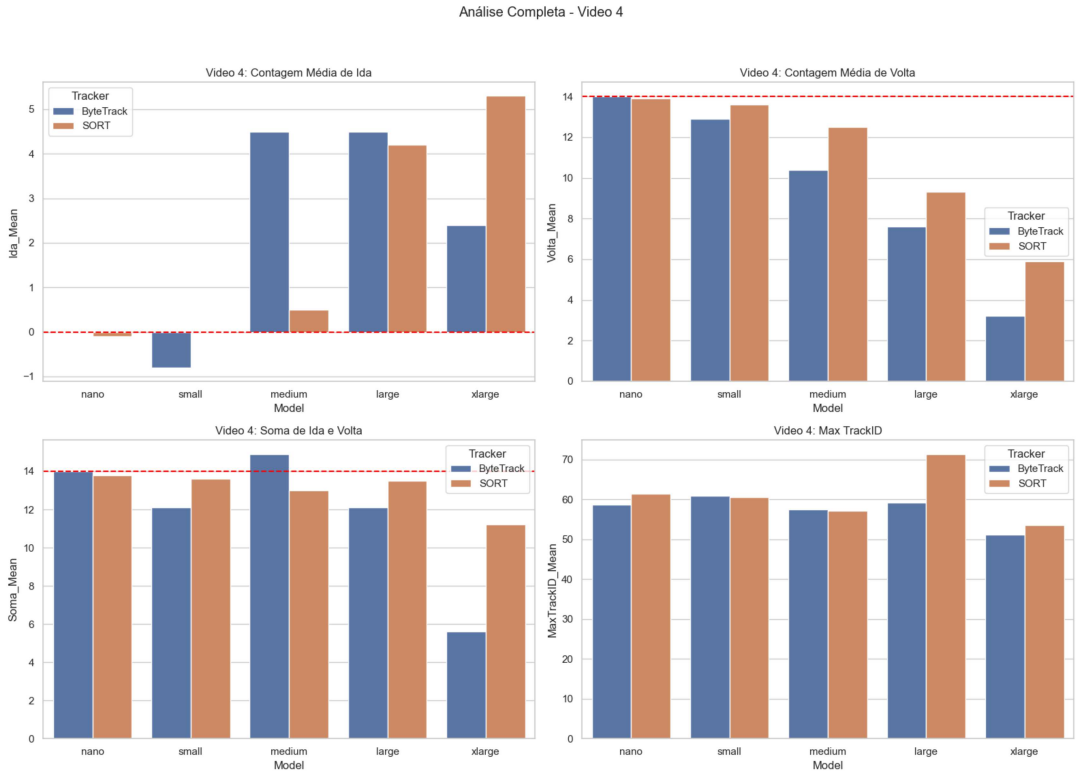


**Figura 13** – Análise completa - Vídeo 4.

No vídeo 4, com 14 animais, o comportamento do ByteTrack com modelo *nano* permaneceu consistente, atingindo resultados próximos aos ideais. O SORT apresentou novamente desempenho inferior, com aumento na contagem de ida e MaxTrackID.

Destaca-se que, neste vídeo, o modelo *medium* do ByteTrack teve desempenho muito próximo ao *nano*, sugerindo que, em cenários com menor quantidade de animais, a diferença entre os tamanhos de modelo se torna menos expressiva.

#### *5.2.1.5* Vídeo 5

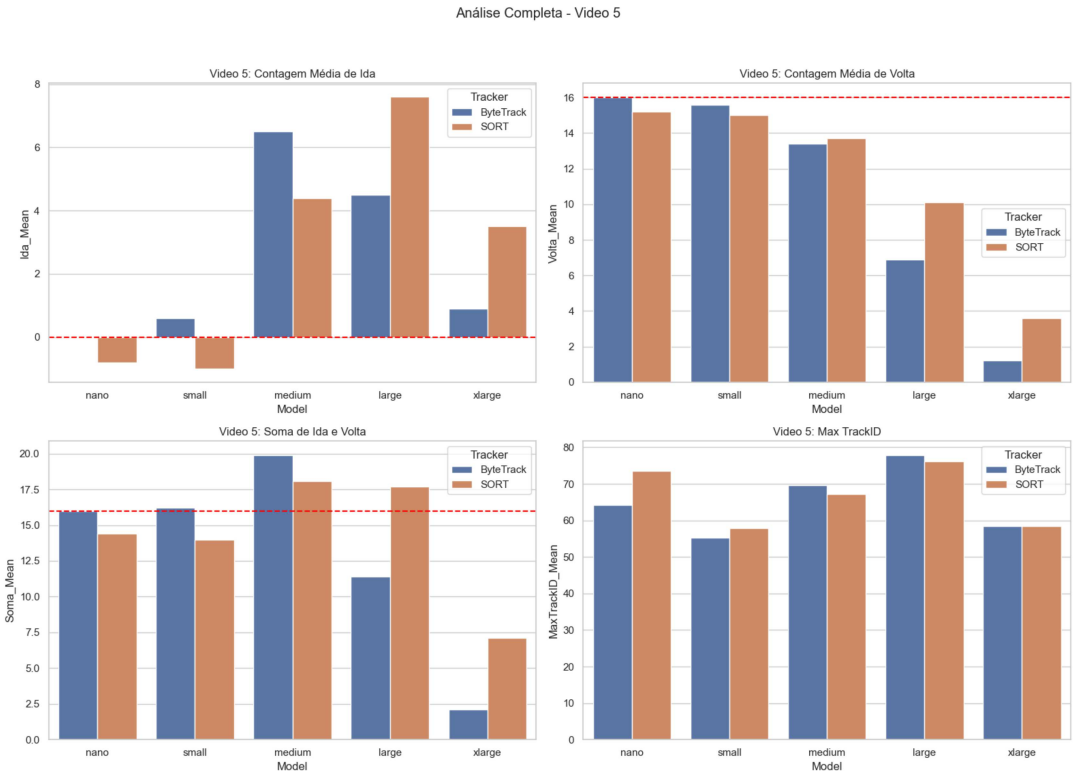


**Figura 14** – Análise completa - Vídeo 5.

O vídeo 5, com 16 animais, destacou novamente a combinação de ByteTrack e modelo *nano*, que obteve a melhor proximidade com os valores ideais em todas as métricas avaliadas. Os modelos superiores apresentaram novamente maior contagem de ida e MaxTrackID, enquanto o SORT repetiu o padrão de baixo desempenho.

#### *5.2.1.6* Vídeo 6

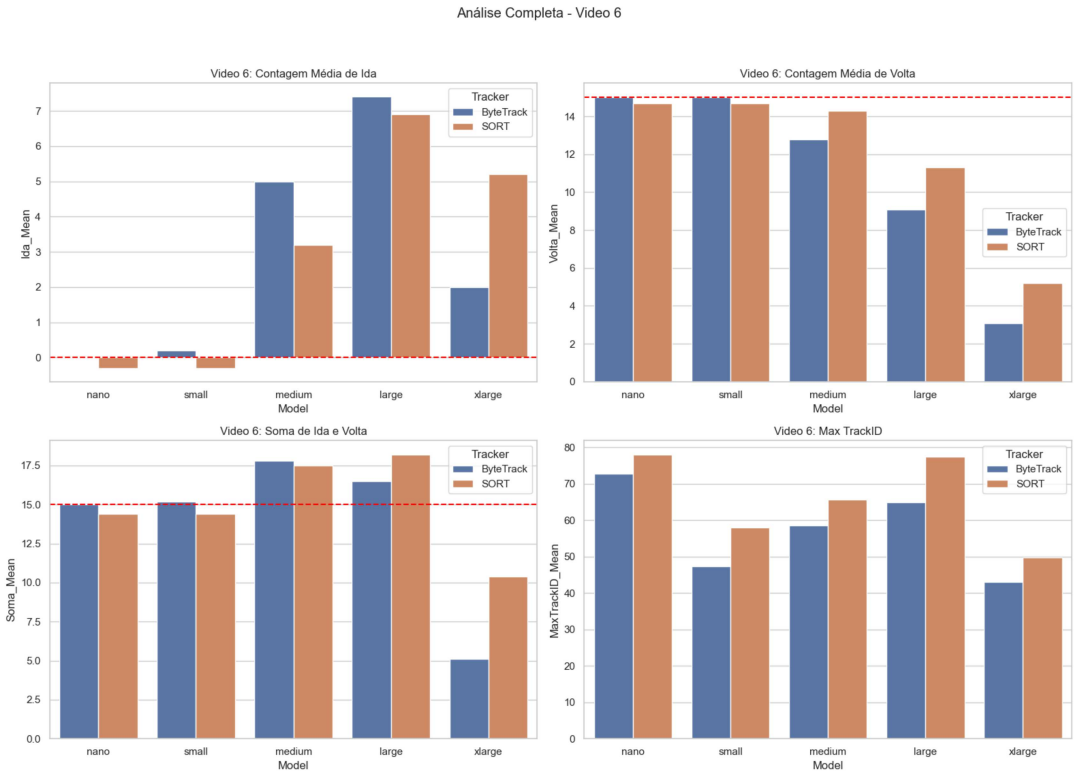


**Figura 15** – Análise completa - Vídeo 6.

No vídeo 6, com 15 animais, os resultados mantiveram a tendência geral. O ByteTrack aliado ao modelo *nano* sustentou o melhor desempenho entre todas as combinações, garantindo baixas contagens de ida e MaxTrackID moderado.

### **5.2.2** Análise geral

A Figura 16 apresenta os resultados agregados das execuções considerando as métricas médias de contagem na ida, na volta, soma total e MaxTrackID por modelo e *tracker*.

Observa-se que o modelo *nano*, aliado ao *tracker* ByteTrack, apresentou o melhor equilíbrio entre as métricas, com uma contagem de ida próxima de zero e uma contagem de volta próxima ao número total de animais, o que era o comportamento esperado considerando a direção principal de movimentação dos suínos no corredor. Além disso, o MaxTrackID médio se manteve relativamente baixo em comparação com outros tamanhos de modelo, evidenciando menor número de rastros criados e maior estabilidade do sistema de rastreamento.

À medida que o tamanho dos modelos aumentou (de *small* a *xlarge*), verificou-se um aumento substancial tanto na contagem média de ida quanto no MaxTrackID, especialmente para o *tracker* SORT.

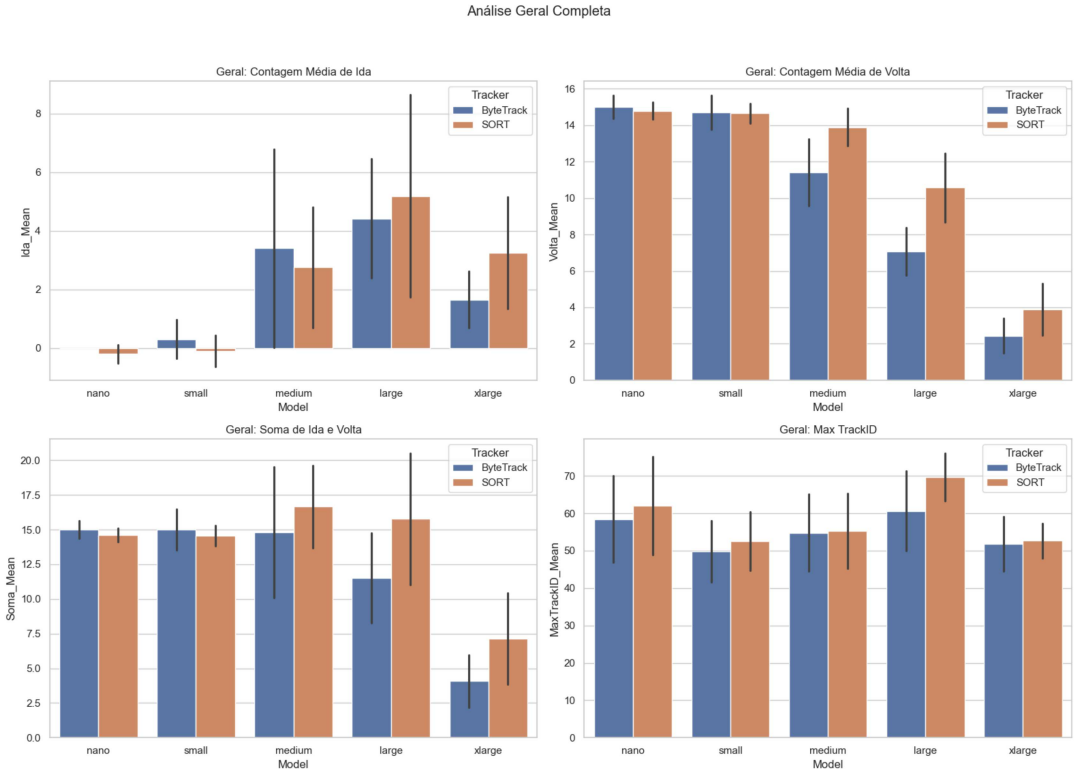


**Figura 16** – Análise geral dos modelos e *trackers*.

O *tracker* SORT apresentou desempenho inferior ao ByteTrack em praticamente todos os cenários, com maiores variações na contagem e valores mais elevados de MaxTrackID, sinalizando menor robustez para a aplicação em questão.

### 5.2.3 Complementação Gráfica: Boxplots e Heatmaps da Análise Geral

Para aprofundar a compreensão da análise geral, foram elaborados boxplots e heatmaps complementares.

Boxplots das Métricas Gerais

A Figura 17 apresenta o boxplot da contagem de ida por modelo e *tracker*. Nota-se que o ByteTrack/*nano* manteve baixa dispersão e ausência de outliers, enquanto outros modelos e o *tracker* SORT apresentaram maior variabilidade.

A Figura 18 ilustra a contagem de volta, onde novamente o ByteTrack *nano* apresentou resultados superiores, mantendo a consistência próxima do valor ideal.

A Figura 19 exibe o MaxTrackID, reforçando a eficiência do ByteTrack, especialmente no modelo *nano*, com menores valores e dispersão limitada.

Por fim, a Figura 20 mostra o tempo total de execução do pipeline. Nota-se que o modelo *nano* para ambos os *trackers* apresentou os menores tempos, com ByteTrack levemente mais eficiente.

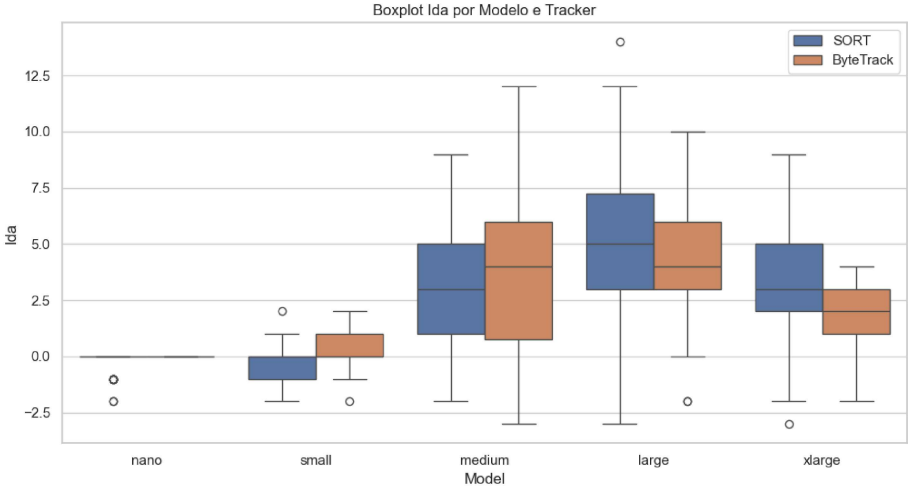


**Figura 17** – Boxplot da contagem de ida por modelo e *tracker*.

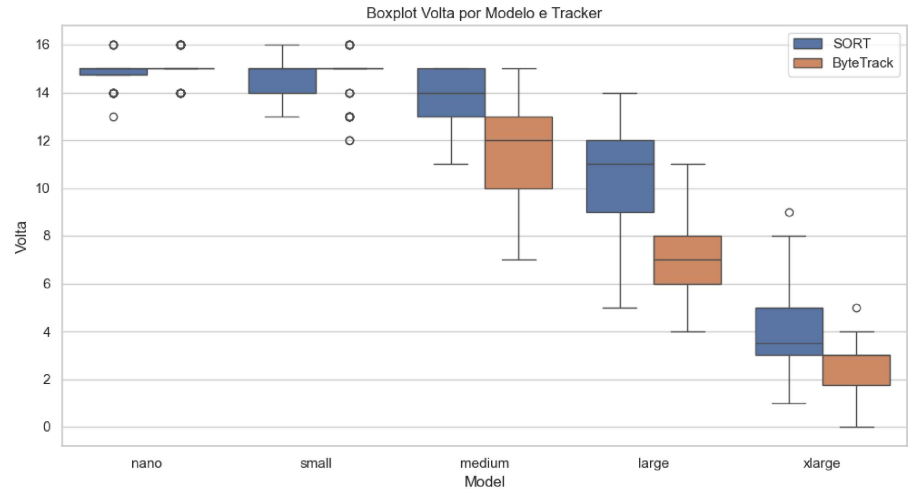


**Figura 18** – Boxplot da contagem de volta por modelo e *tracker*.

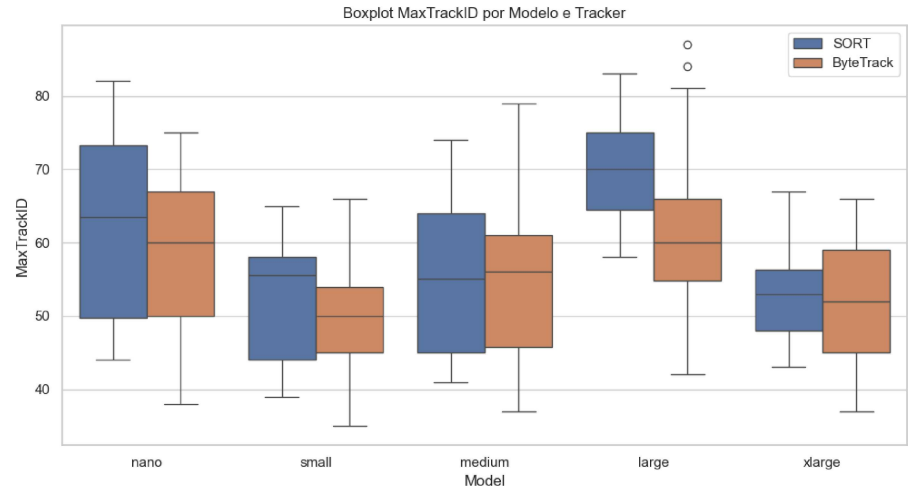


**Figura 19** – Boxplot do MaxTrackID por modelo e *tracker*.

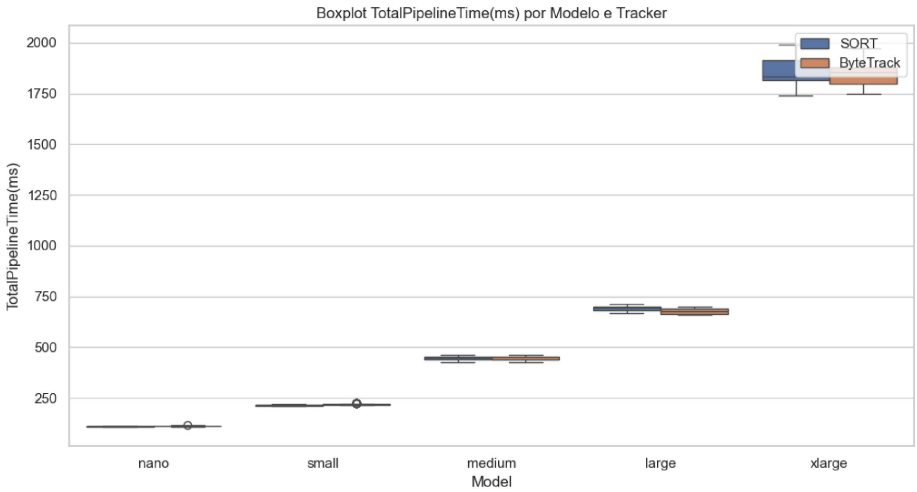


**Figura 20** – Boxplot do tempo total do pipeline por modelo e *tracker*.

Heatmaps dos Desvios Padrão

A Figura 21 apresenta o desvio padrão da contagem de ida. O ByteTrack/*nano* se destacou com desvio zero, evidenciando altíssima consistência.

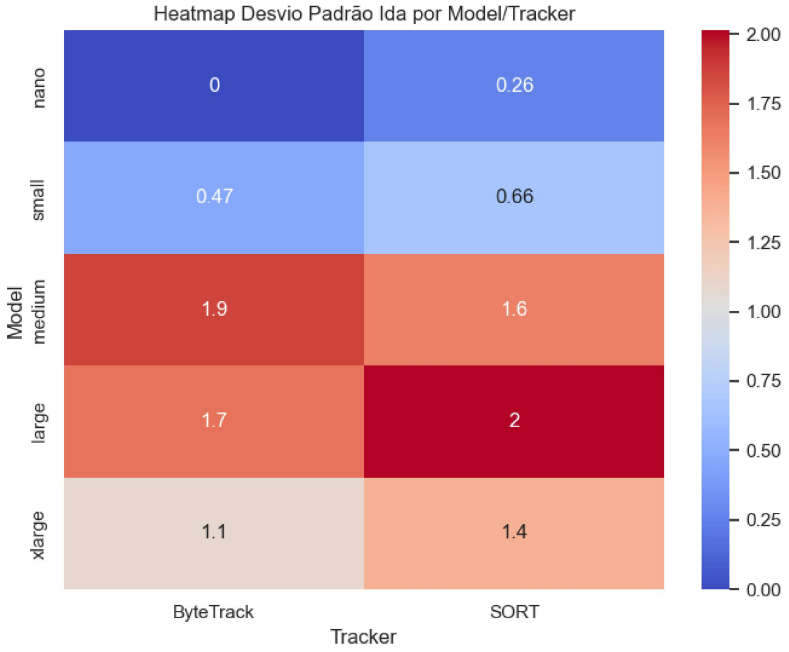


**Figura 21** – Heatmap do desvio padrão da contagem de ida por modelo e *tracker*.

A Figura 22 exibe o desvio padrão da contagem de volta, confirmando a estabilidade do ByteTrack/*nano* em contraste com a maior variabilidade do SORT.

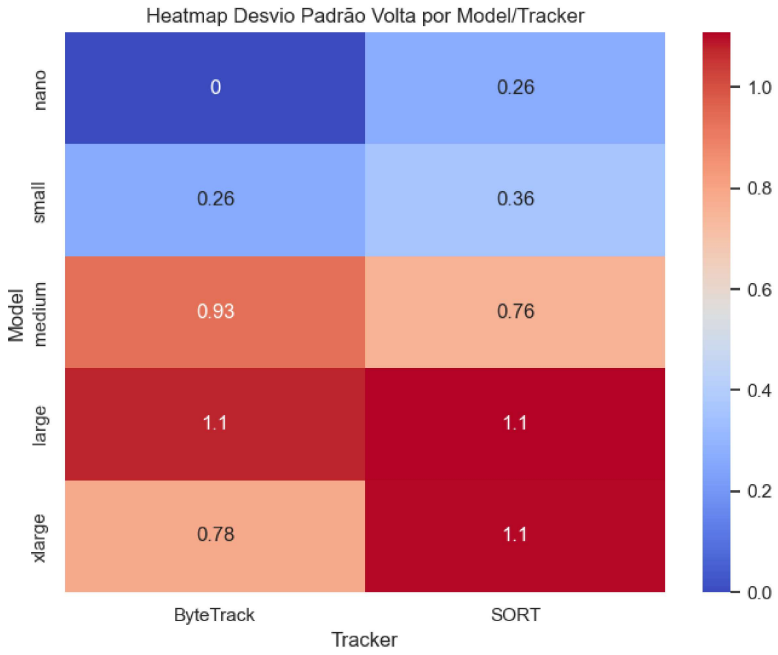


**Figura 22** – Heatmap do desvio padrão da contagem de volta por modelo e *tracker*.

### 5.2.4 Conclusões Parciais da Análise Geral

A análise geral, enriquecida por boxplots e heatmaps, reforça a evidência de que a configuração ByteTrack/*nano* representa a melhor opção para o sistema de contagem de suínos. Esta configuração entregou as melhores médias nas métricas avaliadas, além de demonstrar baixa variabilidade, essencial para aplicações em ambiente real.

As demais combinações de modelo e *tracker* apresentaram desempenho inferior, com maiores desvios nas contagens e aumento expressivo no número de IDs gerados, comprometendo a eficiência do sistema.

## 5.3 Análise de Precisão Final do Algoritmo

Com o objetivo de avaliar a precisão final do sistema desenvolvido para contagem automática de suínos, foi realizada uma análise específica considerando apenas as configurações mais promissoras identificadas nos testes anteriores: SORT e ByteTrack com modelo de identificação nano (SORT/nano e ByteTrack/nano). Foram executadas 600 rodadas de processamento em vídeos diversos, distribuídas entre as diferentes combinações de vídeos e configurações, garantindo robustez estatística para os resultados obtidos.

A Figura 23 apresenta o desempenho dos algoritmos em termos de precisão final, calculada como a razão entre a contagem correta de animais e a contagem real conhecida de cada vídeo analisado.

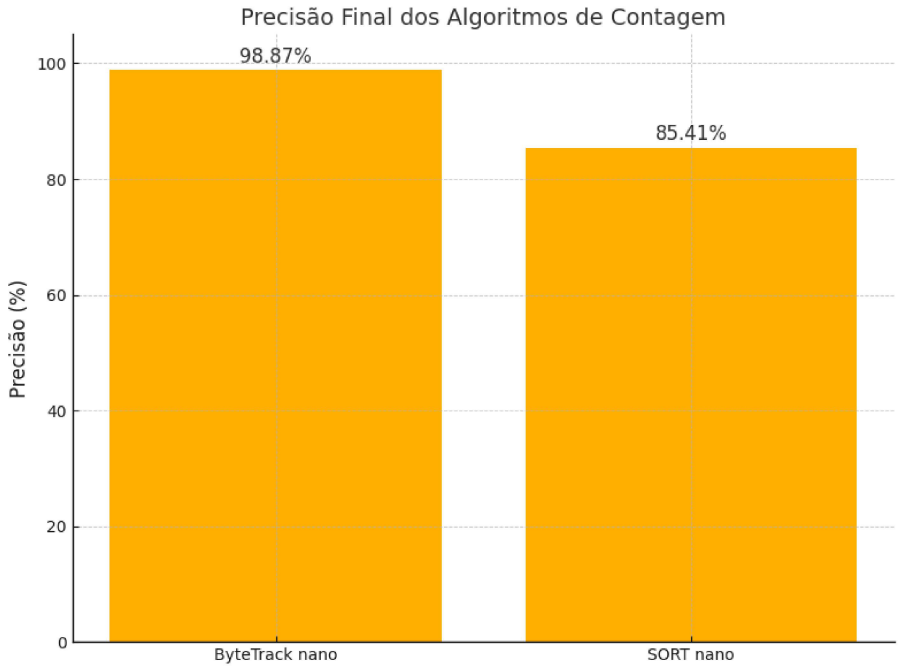


**Figura 23** – Precisão final dos algoritmos ByteTrack/nano e SORT/nano após 600 execuções.

Observa-se que o algoritmo ByteTrack/nano obteve desempenho superior, alcançando uma precisão média de aproximadamente 98,87%, enquanto o algoritmo SORT/nano apresentou uma precisão média de cerca de 85,41%. Este resultado reforça as conclusões das análises anteriores, evidenciando que o ByteTrack/nano se mantém mais estável e eficiente nas diversas condições avaliadas, com melhor capacidade de manter rastros consistentes e reduzir erros de contagem, especialmente em ambientes complexos e com elevada densidade de animais. Este excelente desempenho do ByteTrack/nano demonstra a viabilidade da proposta para aplicações reais na suinocultura de precisão, garantindo alta confiabilidade na contagem automatizada e contribuindo diretamente para a redução de erros operacionais durante o manejo dos animais.

# 6 CONCLUSÃO

Os resultados obtidos ao longo deste trabalho demonstram de forma clara a eficácia da abordagem proposta para a contagem automática de suínos em ambiente de embarcador. Com a realização de **600 execuções experimentais**, foi possível garantir uma base estatística sólida para a validação do sistema, abrangendo diferentes vídeos, condições de iluminação e densidade animal.

Dentre as combinações avaliadas, a utilização do modelo **YOLOv12 nano** aliado ao rastreador **ByteTrack** destacou-se como a melhor configuração, alcançando uma precisão média final de **98,87%**, superando consideravelmente o desempenho do **SORT**, que apresentou uma média de **85,41%**. Estes resultados reforçam a robustez do ByteTrack, que se mostrou mais consistente na manutenção dos rastros e mais tolerante às oclusões e sobreposições frequentes em ambientes confinados.

A análise detalhada das métricas evidenciou não apenas a superioridade do ByteTrack em termos de média de precisão, mas também sua estabilidade ao longo das execuções, com desvios padrão inferiores quando comparados às demais configurações testadas.

Estes achados confirmam a viabilidade prática da solução desenvolvida, principalmente para aplicações em tempo real na cadeia produtiva da suinocultura, oferecendo uma ferramenta confiável para o monitoramento e controle do embarque de animais, contribuindo para a eficiência operacional e para a minimização de erros de manejo.

## 6.1 Trabalhos Futuros

Embora os resultados obtidos sejam extremamente promissores, algumas oportunidades de aprimoramento foram identificadas e devem orientar os trabalhos futuros:

- **Aplicação de redes especializadas para tracking:** A avaliação futura de rastreadores baseados em redes neuronais mais avançadas, como o FairMOT ou o BoT-SORT com otimizações específicas para cenários de alta similaridade visual, pode agregar melhorias adicionais à performance.
- **Desenvolvimento de sistema embarcado completo:** A implementação integral da solução em hardware dedicado, como o NVIDIA Jetson Xavier ou AGX Orin, visando reduzir ainda mais a latência e facilitar a adoção da tecnologia no campo.
- **Expansão para outras espécies e aplicações:** Explorar a aplicação da metodologia para contagem de outras espécies animais ou até mesmo no controle de processos industriais que demandem contagem precisa e em tempo real de objetos em movimento.

- **Aprimoramento da análise temporal:** Incluir modelos que considerem a trajetória histórica dos animais para previsões futuras e detecção antecipada de anomalias no fluxo de movimentação.
- **Melhoria na bufferização do *stream* de vídeo:** Implementar uma estratégia de bufferização dinâmica para o fluxo de vídeo da câmera, permitindo o armazenamento de múltiplos frames de maneira adaptativa. Essa abordagem visa garantir maior robustez na aquisição de imagens, reduzindo a perda de frames em situações de alta carga de processamento ou flutuações na taxa de captura, promovendo melhor sincronia entre captura, inferência e rastreamento.

Com estas direções, espera-se não apenas consolidar o sistema desenvolvido como referência na contagem automática de suínos, mas também ampliar seu impacto para outras áreas da agroindústria e automação de processos, contribuindo para a modernização e eficiência das cadeias produtivas.

# REFERÊNCIAS


1 Gabriel dos Santos Ceretta, Alessandra Matte, and Ana Paula Schervinski Villwock. Circuitos produtivos e a participação da agricultura familiar na suinocultura brasileira: Production circuits and the participation of family farming in brazilian swine farming. *Revista de Geociências do Nordeste*, 11(1):169–179, 2025.

2 O Presente Rural. Projeções para 2025 indicam crescimento modesto na produção e exportação de carne suína. `https://opresenterural.com.br/projecoes-para-2025-indicam-crescimento-modesto-na-producao-e-exportacao-de-carne-suina`, 2025. Acesso em: 20 mar. 2025.

3 Mordor Intelligence. Global pork meat market – analysis and forecast 2023-2030. `https://www.mordorintelligence.com/pt/industry-reports/global-pork-meat-market`, 2023. Acesso em: 20 mar. 2025.

4 3TRES3. Consumo mundial de carne de porco deverá crescer até 2030. `https://www.3tres3.com.pt/ultima-hora/consumo-mundial-de-carne-de-porco-devera-crescer-ate-2030_16660/`, 2023. Acesso em: 20 mar. 2025.

5 CEPEA – Centro de Estudos Avançados em Economia Aplicada. Perspectivas 2025: Suínos – setor deve seguir apostando no mercado externo. `https://www.cepea.esalq.usp.br/br/diarias-de-mercado/perspec-2025-suinos-cepea-em-2025-setor-deve-seguir-apostando-no-mercado-externo.aspx`, 2023. Acesso em: 20 mar. 2025.

6 EMBRAPA. Qualidade da carne suína. `https://www.embrapa.br/qualidade-da-carne/carne-suina`. Acesso em: 20 mar. 2025.

7 Taciana Aparecida DIESEL. *Fatores de risco associados às perdas quantitativas e econômicas ocorridas no manejo pré-abate de suínos*. PhD thesis, Universidade Estadual Paulista "Júlio de Mesquita Filho", Jaboticabal, 2016. 82 f. Tese (Doutorado em Zootecnia) – Faculdade de Ciências Agrárias e Veterinárias. Disponível em: AINFO EMBRAPA. Acesso em: 20 mar. 2025.

8 João Silva and Maria Almeida. Tecnologias de visão computacional na agricultura. *Revista Brasileira de Agricultura Digital*, 4(2):123–135, 2021.

9 Hanse Ahn, Seungwook Son, Heegon Kim, Sungju Lee, Yongwha Chung, and Daihee Park. Ensemblepigdet: Ensemble deep learning for accurate pig detection. *Applied sciences*, 11:5577, 2021. ISSN 2076-3417. doi: 10.3390/app11125577.

10 Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

11 Ali Alameer, Stephanie Buijs, Niamh O'Connell, Luke Dalton, Mona Larsen, Lene Pedersen, and Ilias Kyriazakis. Automated detection and quantification of contact behaviour in pigs using deep learning. *Biosystems engineering*, 224:118–130, 2022. ISSN 1537-5110. doi: 10.1016/j.biosystemseng.2022.10.002.

12 Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

13 Anil Bhujel, Elanchezhian Arulmozhi, Byeong-Eun Moon, and Hyeon-Tae Kim. Deep-learning-based automatic monitoring of pigs' physico-temporal activities at different greenhouse gas concentrations. *Animals (Basel)*, 11:3089, 2021. ISSN 2076-2615. doi: 10.3390/ani11113089.

14 Eric T Psota, Mateusz Mittek, Lance C Pérez, Ty Schmidt, and Benny Mote. Multi-pig part detection and association with a fully-convolutional network. *Sensors (Basel)*, 19:852, 2019. ISSN 1424-8220. doi: 10.3390/s19040852.

15 Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

16 Shuqin Tu, Weijun Yuan, Yun Liang, Fan Wang, and Hua Wan. Automatic detection and segmentation for group-housed pigs based on pigms r-cnn. *Sensors (Basel, Switzerland)*, 21:3251, 2021. ISSN 1424-8220. doi: 10.3390/s21093251. This manuscript is extension version of the conference paper: Tu, S. Liu, H. Li, J. Huang, J. Li, B. Pang, J. Xue, Y. Instance Segmentation Based on Mask Scoring R-CNN for Group-housed Pigs. In Proceedings of the 2020 International Conference on Computer Engineering and Application (ICCEA 2020), Guangzhou, China, 27–29 March 2020.

17 Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6409–6418, 2019.

18 Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

19 Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017.

20 Xiaopin Wang, Wei Wang, Jisheng Lu, and Haiyan Wang. Hrst: An improved hrnet for detecting joint points of pigs. *Sensors (Basel, Switzerland)*, 22:7215, 2022. ISSN 1424-8220. doi: 10.3390/s22197215.

21 Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

22 Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

23 Marko Ocepek, Anja Žnidar, Miha Lavrič, Dejan Škorjanc, and Inger Lise Andersen. Digipig: First developments of an automated monitoring system for body, head and

tail detection in intensive pig farming. *Agriculture (Basel)*, 12:2, 2021. ISSN 2077-0472. doi: 10.3390/agriculture12010002.

24 Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

25 Weixian Song, Junlong Fang, Runtao Wang, and Kezhu Tan. Detection of pig based on improved resnet model in natural scene. *Applied mathematics and nonlinear sciences*, 6:215–226, 2021. ISSN 2444-8656. doi: 10.2478/amns.2021.2.00040.

26 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

27 Zishun Zhou. Detection and counting method of pigs based on yolov5_plus: A combination of yolov5 and attention mechanism. *Mathematical problems in engineering*, 2022:1–16, 2022. ISSN 1024-123X. doi: 10.1155/2022/7078670.

28 Glenn Jocher. YOLOv5 by Ultralytics, 5 2020. URL `https://github.com/ultralytics/yolov5`.

29 Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

30 Martin Wutke, Felix Heinrich, Pronaya Prosun Das, Anita Lange, Maria Gentz, Imke Traulsen, Friederike K Warns, Armin Otto Schmitt, and Mehmet Gültas. Detecting animal contacts—a deep learning-based pig detection and tracking approach for the quantification of social contacts. *Sensors (Basel, Switzerland)*, 21:7512, 2021. ISSN 1424-8220. doi: 10.3390/s21227512.

31 E T Psota, T Schmidt, B Mote, and L C Pérez. Long-term tracking of group-housed livestock using keypoint detection and map estimation for individual animal identification. *Sensors (Switzerland)*, 20:1–25, 2020. doi: 10.3390/s20133670.

32 Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

33 Jake Cowton, Ilias Kyriazakis, and Jaume Bacardit. Automated individual pig localisation, tracking and behaviour metric extraction using deep learning. *IEEE access*, 7:108049–108060, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2933060.

34 Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

35 Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

36 Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

37 N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017.

38 Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*, pages 17–35. Springer, 2016.

39 D Xiao, A Feng, and J Liu. Detection and tracking of pigs in natural environments based on video analysis. *International Journal of Agricultural and Biological Engineering*, 12:116–126, 2019. doi: 10.25165/ijabe.v12i4.4591. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85071689009&doi=10.25165%2fijabe.v12i4.4591&partnerID=40&md5=aa8902899cb0b245af8905bdc76ee362`.

40 W Gong, J Wang, L Mao, and L Lu. A pig tracking algorithm with improved iou-tracker. *Proceedings of SPIE - The International Society for Optical Engineering*, 12349, 2022. doi: 10.1117/12.2657508. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85141868566&doi=10.1117%2f12.2657508&partnerID=40&md5=4f505b039979b3b286a682821a8cc0f2`. Export Date: 27 January 2023 | RAYYAN-INCLUSION: "Luan"=>"Included".

41 Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*, pages 1–6. IEEE, 2017.

42 Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819, 2011.

43 Jonggwan Kim, Yooil Suh, Junhee Lee, Heechan Chae, Hanse Ahn, Yongwha Chung, and Daihee Park. Embeddedpigcount: Pig counting with video object detection and tracking on an embedded board. *Sensors (Basel)*, 22:2689, 2022. ISSN 1424-8220. doi: 10.3390/s22072689.

44 Mengxiao Tian, Hao Guo, Hong Chen, Qing Wang, Chengjiang Long, and Yuhao Ma. Automated pig counting using deep learning. *Computers and electronics in agriculture*, 163:104840, 2019. ISSN 0168-1699. doi: 10.1016/j.compag.2019.05.049.

45 Daniel Onoro-Rubio and Roberto J López-Sastre. Towards perspective-free object counting with deep learning. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*, pages 615–629. Springer, 2016.

46 Y Hu, Y Cang, and Y Qiao. Design of intelligent pig counting system based on improved instance segmentation algorithm. *Nongye Gongcheng Xuebao/Transactions of the Chinese Society of Agricultural Engineering*, 36:177–183, 2020. doi: 10.11975/j.issn.1002-6819.2020.19.020. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85098264423&doi=10.11975%2fj.issn.1002-6819.2020.19.020&partnerID=40&md5=b9fa38ee46a7ebf6c5018cf669d39b8f`.

47 Chengqi Liu, Jie Su, Longhe Wang, Shuhan Lu, and Lin Li. La-deeplab v3+: A novel counting network for pigs. *Agriculture (Basel)*, 12:284, 2022. ISSN 2077-0472. doi: 10.3390/agriculture12020284.

48 Yongliang Qiao, Matthew Truman, and Salah Sukkarieh. Cattle segmentation and contour extraction based on mask r-cnn for precision livestock farming. *Computers and electronics in agriculture*, 165:104958, 2019. ISSN 0168-1699. doi: 10.1016/j.compag.2019.104958.

49 Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 75–91. Springer, 2016.

50 Pedro O O Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. *Advances in neural information processing systems*, 28, 2015.

51 Min Jiang, Yuan Rao, Jingyao Zhang, and Yiming Shen. Automatic behavior recognition of group-housed goats using deep learning. *Computers and electronics in agriculture*, 177:105706, 2020. ISSN 0168-1699. doi: 10.1016/j.compag.2020.105706.

52 Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

53 Wei Li, Yang Xiao, Xibin Song, Na Lv, Xinbo Jiang, Yan Huang, and Jingliang Peng. Chicken image segmentation via multi-scale attention-based deep convolutional neural network. *IEEE access*, 9:61398–61407, 2021. ISSN 2169-3536. doi: 10.1109/ACCESS.2021.3074297.

54 Chongxin Tao, Yizhuo Meng, Junjie Li, Beibei Yang, Fengmin Hu, Yuanxi Li, Changlu Cui, and Wen Zhang. Msnet: multispectral semantic segmentation network for remote sensing images. *GIScience & Remote Sensing*, 59(1):1177–1198, 2022.

55 Zhihui Tian, Xiaoyu Guo, Xiaohui He, Panle Li, Xijie Cheng, and Guangsheng Zhou. Mscanet: multiscale context information aggregation network for tibetan plateau lake extraction from remote sensing images. *International Journal of Digital Earth*, 16(1): 1–30, 2023.

56 Xier Chen, Yanchao Lian, Licheng Jiao, Haoran Wang, YanJie Gao, and Shi Lingling. Supervised edge attention network for accurate image instance segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pages 617–631. Springer, 2020.

57 S Sun, J Qin, and H Xue. Sheep delivery scene detection based on faster-rcnn. *Proceedings of SPIE - The International Society for Optical Engineering*, 11321, 2019. doi: 10.1117/12.2538904. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85077815709&doi=10.1117%2f12.2538904&partnerID=40&md5=6e2f41d18be3d4f5c6c82480203321b1`.

58 Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

59 Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

60 Yunjie Tian, Qixiang Ye, and David Doermann. Yolov12: Attention-centric real-time object detectors, 2025. Technical Report.

61 NVIDIA Corporation. Jetson nano developer kit technical specifications. `https://developer.nvidia.com/embedded/jetson-nano-developer-kit`, 2024. Accessed: 20 Mar. 2025.

62 Xiao Zhou, Xinjing Wang, et al. Bytetrack: Multi-object tracking by associating every detection box, 2021. arXiv preprint.

63 Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

64 Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

65 Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

66 Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

67 Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

68 Chenggang Li et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022. URL `https://arxiv.org/abs/2209.02976`.

69 Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022. URL `https://arxiv.org/abs/2207.02696`.

70 Ultralytics. YOLOv8 by Ultralytics, 2023. URL `https://github.com/ultralytics/ultralytics`. Documentação oficial disponível em `https://docs.ultralytics.com/`.

71 Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *arXiv preprint arXiv:2402.13616*, 2024. URL `https://arxiv.org/abs/2402.13616`.

72 Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. YOLOv10: Real-Time End-to-End Object Detection. *arXiv preprint arXiv:2405.14458*, 2024. URL `https://arxiv.org/abs/2405.14458`.

73 Rahima Khanam and Muhammad Hussain. YOLOv11: An Overview of the Key Architectural Enhancements. *arXiv preprint arXiv:2410.17725*, 2024. URL `https://arxiv.org/abs/2410.17725`.

74 Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

75 Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. In *arXiv preprint arXiv:1603.00831*, 2016. URL `https://motchallenge.net`.

76 P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. In *arXiv preprint arXiv:2003.09003*, 2020. URL `https://motchallenge.net`.

77 Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

78 Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2017.

79 K Zhang, D Li, J Huang, and Y Chen. Automated video behavior recognition of pigs using two-stream convolutional networks. *Sensors (Switzerland)*, 20, 2020. doi: 10.3390/s20041085. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85079573495&doi=10.3390%2fs20041085&partnerID=40&md5=76397d6484ada3c46f9549c6458c826a`.

80 Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018.

81 Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

82 Zhun Zhong, Liang Zheng, Guolei Li, et al. Random erasing data augmentation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

83 Paulius Micikevicius, Shubho Narang, Jonah Alben, et al. Mixed precision training. In *International Conference on Learning Representations (ICLR)*, 2018.

84 Roy Jonker and Ton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*, pages 325–331. Springer Berlin Heidelberg, 1988.